
Universidade de Aveiro
2008
Departamento de Electrónica, Telecomunicações e Informática

Manuel Joaquim da Silva Almeida

Métodos e Ferramentas para Reconfiguração de FPGAs Remotamente


Universidade de Aveiro
2008
Departamento de Electrónica, Telecomunicações e Informática

Manuel Joaquim da Silva Almeida

Métodos e Ferramentas para Reconfiguração de FPGAs Remotamente

dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Electrónica e Telecomunicações, realizada sob a orientação científica do Prof. Dr. Valeri Skliarov, Professor Catedrático do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

o júri

presidente

Prof. Doutor António Rui de Oliveira e Silva Borges
professor associado da Universidade de Aveiro

Prof. Doutor Valeri Skliarov
professor catedrático da Universidade de Aveiro (Orientador)

Prof. Doutor Hélio Sousa Mendonça
professor associado da Faculdade de Engenharia da Universidade do Porto

Prof. Doutor Arnaldo Silva Rodrigues de Oliveira
professor auxiliar convidado da Universidade de Aveiro

agradecimentos

Cabe-me aqui expressar a minha profunda e sincera gratidão a todos aqueles que, de algum modo, colaboraram comigo ao longo deste trabalho.

Ao Professor Valeri Skliarov, meu Orientador Científico, agradeço a disponibilidade demonstrada, as condições que me foram proporcionadas na realização deste trabalho e a motivação constante ao longo desta caminhada/percurso. Agradeço, também, por todos os conhecimentos que me foram transmitidos, pelo acompanhamento e revisão atenta concedida a esta tese.

Agradeço ao meu colega do Laboratório de Sistemas Computacionais, Bruno Pimentel pelo trabalho realizado, por todas as sugestões e considerações.

Aos meus irmãos, sobrinhos e de uma forma muito especial aos meus pais pela ajuda, pelo suporte emocional e encorajamento que me deram.

Aos meus amigos, a compreensão e força transmitida foram fundamentais para a concretização desta meta e pelos sempre surpreendentes convívios em que pude desfrutar da companhia deles.

A todos os que de uma forma directa ou indirecta me ajudaram a terminar esta tese, o meu profundo reconhecimento e agradecimento.

palavras-chave

Sistemas Reconfiguráveis, FPGA, Placas de Prototipagem, Maquina de Estados Finitos, Placa Expansão.

resumo

O trabalho foi realizado no âmbito dos sistemas digitais reconfiguráveis. Do trabalho desenvolvido o resultado principal é a placa de prototipagem que utiliza uma FPGA da família Spartan-3 da Xilinx. A placa tem uma série de características distintas, das quais se realçam: configuração / interacção modificável (com cabo - USB ou sem cabo - Bluetooth) com um dispositivo anfitrião (por exemplo um PC); um custo relativamente baixo; expansibilidade através de um grande número de extensões para ser concebida e implementada em hardware. Foram implementados alguns projectos para demonstrar as capacidades de reconfiguração, e permitir a funcionalidade de dispositivos, tais como máquinas de estados finito, a ser alterado durante o tempo de execução.

keywords

Reconfigurable Systems, FPGA, Prototyping Boards, Finite State Machine, Expantion Boards.

abstract

The work is done in the scope of reconfigurable digital systems with primary focus on prototyping. The major result of the work is the developed prototyping board with a core FPGA of the Xilinx Spartan-3 family. The board has a number of distinctive features, the most important of which are: changeable (wired - USB or wireless - Bluetooth) configuration/interaction with a host device (such as a PC); a relatively low cost; and expandability through a vast variety of extensions to be designed and implemented in hardware. To illustrate capabilities for dynamic reconfiguration some projects have been implemented enabling the functionality of prevalent devices, such as finite state machines, to be changed during run-time.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ASIC	Application-Specific Integrated Circuit
CAD	Computer Aided Design
DAC	Digital to Analog Converter
EEPROM	Electrically Erasable Programmable Read-Only Memory
FIFO	First In, First Out
FPGA	Field-Programmable Gate Array
HT	Hardware Templates
LCD	Liquid Crystal Display
LED	Light Emitting Diode
MRAM	Multiplexer Random Access Memory
MEF	Máquina de Estados Finitos
PBM	Prototyping Board Manager
PLD	Programmable Logic Device
PM	Programmable Multiplexer
RAM	Random Access Memory
ROM	Read Only Memory
STRAM	Static Transition Random Access Memory
VGA	Video Graphics Array
VHDL	VHSIC Hardware Description Language
VHSIC	Very-High-Speed Integrated Circuit
LECT	Licenciatura em Engenharia de Computadores e Telemática
MIECT	Mestrado Integrado em Engenharia de Computadores e Telemática
LEET	Licenciatura em Engenharia Electrónica e Telecomunicações
MIEET	Mestrado Integrado em Engenharia Electrónica e Telecomunicações

Capítulo 1

Introdução

Sumário

Neste capítulo são apresentadas as motivações e estabelecem-se os objectivos principais do trabalho. É também feita uma introdução à organização da tese.

1.1 Motivação

Os sistemas reconfiguráveis existentes no mercado apresentam, ainda, valores de difícil acesso económico. Apesar de se apresentar com enormes potencialidades, não tem tido a divulgação necessária para permitir o seu crescimento de forma mais rápida. No que se refere aos preços das placas de prototipagem, no momento em que se iniciou este trabalho, os valores eram bastantes elevados. Esta é uma das razões que faz com que o acesso seja mais difícil, sendo então, complicada a inserção em grande número destes sistemas em empresas e no processo educativo. Observando a conjuntura da altura, idealizou-se uma placa que possibilitasse combater os problemas existentes.

A existência de um grande número de periféricos disponíveis, torna difícil obter um conhecimento profundo da maioria deles. Como a maioria dos sistemas reconfiguráveis existentes no mercado têm dois ou três periféricos disponíveis é necessário implementar interfaces que permitam a sua integração. Devido à carência de pinos de entrada/saída disponibilizados pelos fabricantes destes sistemas, por vezes, não existe a possibilidade de ligação dessas mesmas interfaces.

A existência de modelos de *Hardware Template* (HT) [1] possibilita a implementação de métodos para síntese de Máquinas de Estados Finitos (MEF). Estas permitem implementar diferentes funcionalidades, que se traduzem na resolução de um grande número de problemas.

1.2 Objectivos

Os objectivos principais deste trabalho são os seguintes:

- Desenvolvimento de uma placa de prototipagem utilizando uma FPGA Spartan-3 que pode ser configurada através da porta USB ou do módulo *Bluetooth*;
- Concepção de métodos e ferramentas para configuração da FPGA;
- Implementação de máquinas de estados finitos reconfiguráveis remotamente baseadas em HT, que é um circuito com uma estrutura predefinida que já foi implementado em hardware;
- Desenvolvimento de placas de extensão, para integração na placa de prototipagem desenvolvida, para resolver vários problemas em engenharia.

1.3 Organização da tese

Esta tese está dividida em sete capítulos. O capítulo 2 apresenta uma perspectiva histórica do desenvolvimento dos sistemas reconfiguráveis, dando um maior ênfase às FPGAs e placas de prototipagem.
O capítulo 3 descreve todo o processo de desenvolvimento da placa de prototipagem DETIUA-S3.
Métodos e ferramentas para reconfiguração da FPGA são apresentados no capítulo 4. Este capítulo vem em sequência do anterior.
O capítulo 5 é dedicado a máquinas de estados finitos reconfiguráveis remotamente.
Extensões da placa de prototipagem para resolver problemas em engenharia são o tema do capítulo 6. Neste capítulo são apresentados alguns módulos/placas de extensão, que fazem a interface necessária para a integração de periféricos na placa DETIUA-S3.

Capítulo 2

Sistemas Reconfiguráveis

Sumário

Os Dispositivos Lógicos Programáveis têm evoluído de tal modo, que têm forçado o mercado a transformar-se. Este facto tem facilitado o crescimento das PLDs nos sistemas digitais. A evolução rápida das características das PLDs está a permitir a introdução de novas funcionalidades e a criação de novos métodos por parte dos projectistas.

Este capítulo apresenta uma resenha da evolução dos sistemas reconfiguráveis, mais concretamente das FPGAs e das placas de prototipagem utilizadas nos últimos anos na Universidade de Aveiro, mais precisamente no departamento de electrónica, telecomunicações e informática.

2.1 Introdução

Actualmente os dispositivos reconfiguráveis são considerados como uma alternativa aos ASICs, e são já utilizados com sucesso num grande número de aplicações práticas. Estes constituem uma ferramenta rápida e versátil para prototipagem de hardware, para o desenvolvimento de novas soluções, para o auxílio em tarefas computacionalmente intensivas e ainda para explorar e testar projectos no processo educativo.

Vários fabricantes têm vindo a construir novas placas de prototipagem baseadas em FPGAs, que comummente integram um elevado número de interfaces e dispositivos periféricos fazendo aumentar o respectivo preço. A aplicabilidade destas placas é assim alargada em diversas áreas, tais como processamento de imagem, optimização combinatória, multimédia, etc.

2.2 Evolução dos sistemas reconfiguráveis

Os sistemas reconfiguráveis prometem vir a ser preponderantes em muitos dos sistemas computacionais do futuro. A utilização de sistemas reconfiguráveis tem vindo a ser cada vez maior, sendo por vezes a única forma de atingir, com sucesso, características como: alto desempenho, flexibilidade, baixo consumo de energia, etc.

Tabela 2.1 – Eventos históricos importantes

Ano	**Eventos Importantes**
1947	Shockley, *et al*. – o primeiro transistor nos laboratórios Bell
1958	Jack Kilby – o primeiro circuito integrado
1959-1960	G. Estrin, origens dos computadores reconfiguráveis
1962	Hofstein, *et al*. – transistor de efeito de campo de semicondutor metal-óxido (MOSFET)
1970	Intel – a primeira DRAM de 1024-bit
1970	Fairchild – a primeira SRAM 256-bit
1970	Ron Cline (Signetics) – a primeira PLD (PLA)
1971	Intel – o primeiro microprocessador, 4004
1978	Monolithic Memories Inc. – o primeiro PAL
1984	Altera – o primeiro CPLD baseado na combinação da tecnologia CMOS e EPROM
1985	Xilinx – a primeira FPGA, XC2064™ - uma nova forma de lógica programável
1991	O primeiro computador reconfigurável comercial do mundo, o Algotronix CHS2X4

Na Tabela 2.1 encontra-se um sumário de eventos importantes que aconteceram desde o aparecimento do primeiro transístor [2].

Os dispositivos de lógica programável (PLDs) podem ser considerados "plataforma

virgem para projectos digitais", à qual pode ser dada uma funcionalidade apropriada para um dado equipamento electrónico em que está inserido. Estas aplicações podem ser tão diversas como: sistemas para automóveis; imagem; telecomunicações, etc. Isto significa que o grande custo associado ao desenvolvimento de dispositivos semicondutores pode ser repartido ao longo da cadeia de clientes (ao contrário do processador normal em que os encargos são suportados pelo cliente, para quem foi projectado). Não só os clientes podem beneficiar da economia de escala, como o silício por si só, não está ligado a um mercado final existindo menor probabilidade de ser considerado supérfluo ou obsoleto. As mesmas FPGAs são fornecidas para automóveis, consumidor final, telecomunicações, e outros mercados. Isto significa que os PLDs não são tão afectados pelo ciclos finais de mercados, que historicamente têm demonstrado ter picos em diferentes momentos.

Para reduzir os custos globais do produto, um dos caminhos utilizados por muitas empresas, é o de dedicar algum do seu tempo para reduzir a média de componentes utilizados e reduzir, na lista de materiais, o número de dispositivos. A integração de projecto, ou seja, o processo de redução de componentes colocando um sistema eficaz, é um desses métodos. Actualmente, isso pode ser alcançado através da integração de muitos dispositivo num único ASIC, tentando colocar, se possível, toda a funcionalidade em *software* ou integrando várias funções numa FPGA. Tal como foi referido anteriormente, podemos ver que: enquanto no passado a escolha do ASIC foi uma escolha natural, este caminho é agora reservado para aqueles que têm uma grande produção em funcionamento e que não são afectados pelas pressões do mercado. Colocar funções em *software* é agradável mas, em alguns casos, demora muito tempo a testar e depurar. Muitas vezes, o tempo pode não ser utilizado nesta altura de testes mas pode ser na implementação do projecto em si. Agora que o custo do silício da FPGA se aproxima do equivalente ao ASIC, muitas empresas estão a preferir integrar os seus projectos em FPGAs.

No geral, a integração de projectos permite:

- Redução da lista de material;
- Redução nas despesas da qualidade dos dispositivos;
- Redução no custo dos inventários e da gestão;

- Redução nos custos de stock;
- Pode reduzir a complexidade do PCB e, consequentemente, o custo do PCB.

A integração de projectos em FPGAs também permite:

- Menor tempo de colocação no mercado;
- Mesmo PCB para muitos projectos;
- Flexibilidade para mudar projectos em qualquer fase – mesmo no campo;
- Controlo do custo de fim de vida.

Os custos de um projecto estão sobretudo relacionados com a mão-de-obra e não com custos recorrentes de engenharia [3]. Projectos em PLDs necessitam de menor mão-de-obra (ver Figura 2.1), porque as alterações do projecto e de depuração podem ser efectuadas instantaneamente. O hardware e software podem ser desenvolvidos ao mesmo tempo, sendo desnecessária a criação de um vector de teste, e o tempo de preparação do produto será praticamente nulo, em comparação com os ASICs. Os PLDs não têm custos externos recorrentes de engenharia, o que é um aspecto importante, já que os volumes de produção são baixos nos mercados de evolução rápidos. Em muitos casos, as ferramentas disponibilizadas pelas empresas de PLDs são de acesso livre, aumentando a competitividade em termos económicos.

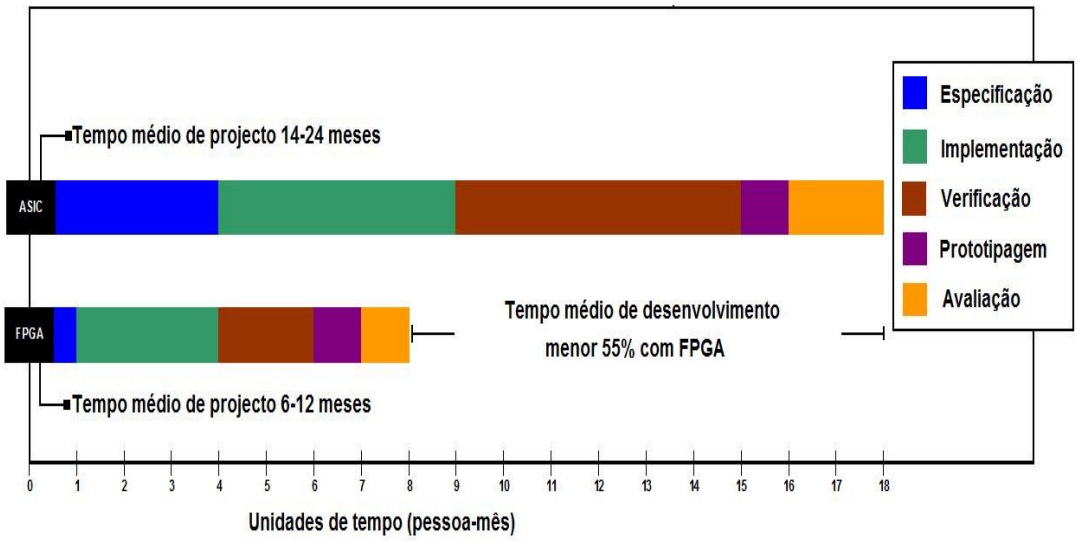

Figura 2.1 – Tempo de projecto da FPGA vs. ASIC [3] – Microprocessador de uso geral

Na Figura 2.2 pode-se verificar a cota de mercado das principais empresas que fabricam PLDs (Xilinx – 51%, Altera – 32%, Lattice – 8%, Actel – 6%, QuickLogic – 2%, Outras – 1%).

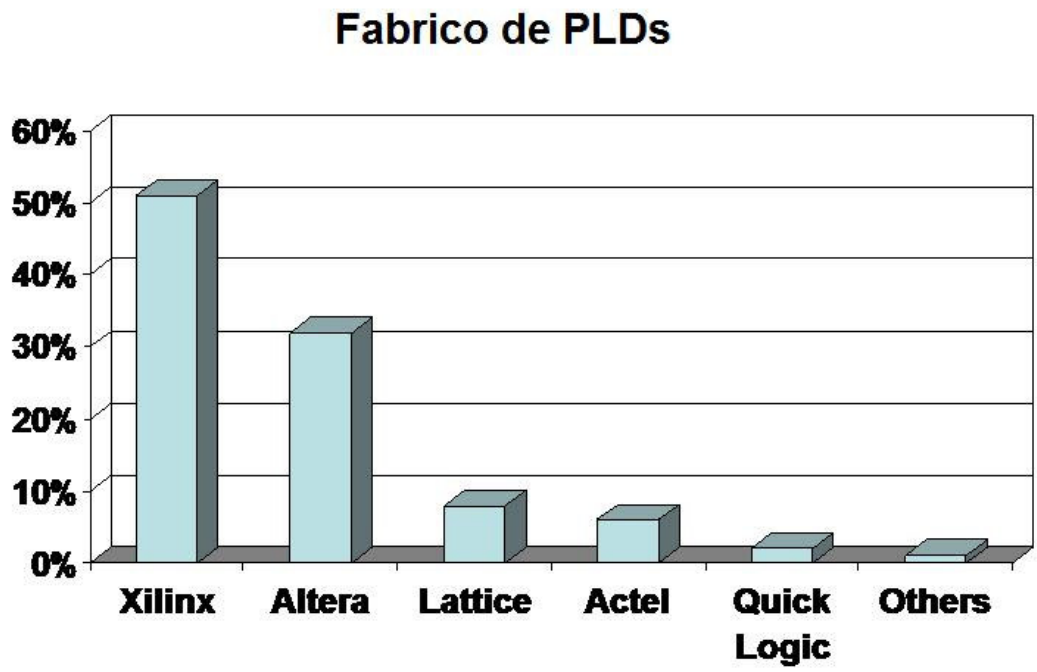

Figura 2.2 – Fabrico de PLDs

Os fabricantes produzem FPGAs, de dois em dois anos, alcançando normalmente o dobro da densidade, com um melhor desempenho (ver Figura 2.3) [2, 4]. Este facto está em conformidade com a lei de Moore [5], já que Gordon Moore, o fundador da Intel, constatou que a cada 2 anos, a capacidade de processamento dos computadores dobra, enquanto os custos permanecem constantes [6].

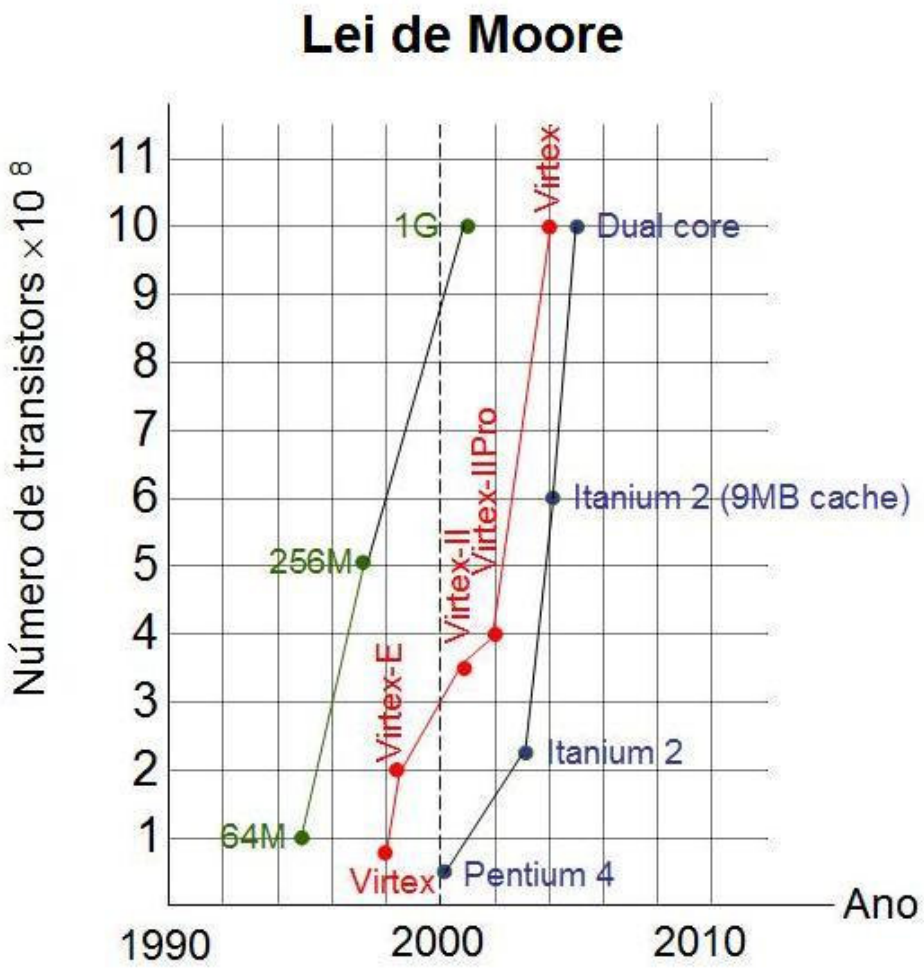

Figura 2.3 – FPGAs permitem utilizar lei de Moore

2.3 Evolução das FPGAs

Tal como foi referido anteriormente, as FPGAs têm sofrido uma grande evolução ao longo dos anos. Como se pode verificar através da Figura 2.2, existem várias empresas que se dedicam à produção de FPGAs, e portanto seria extremamente exaustivo descrever a evolução das FPGAs de todos esses fabricantes [7, 8]. Por isso, será dado um maior ênfase às FPGAs da Xilinx, que de momento é a maior empresa do sector.

Na Figura 2.4 é possível fazer uma análise às características das FPGAs da Xilinx, que apresenta duas grandes famílias: Spartan e Virtex. É possível verificar a evolução das FPGAs dentro da mesma família e em famílias diferentes.

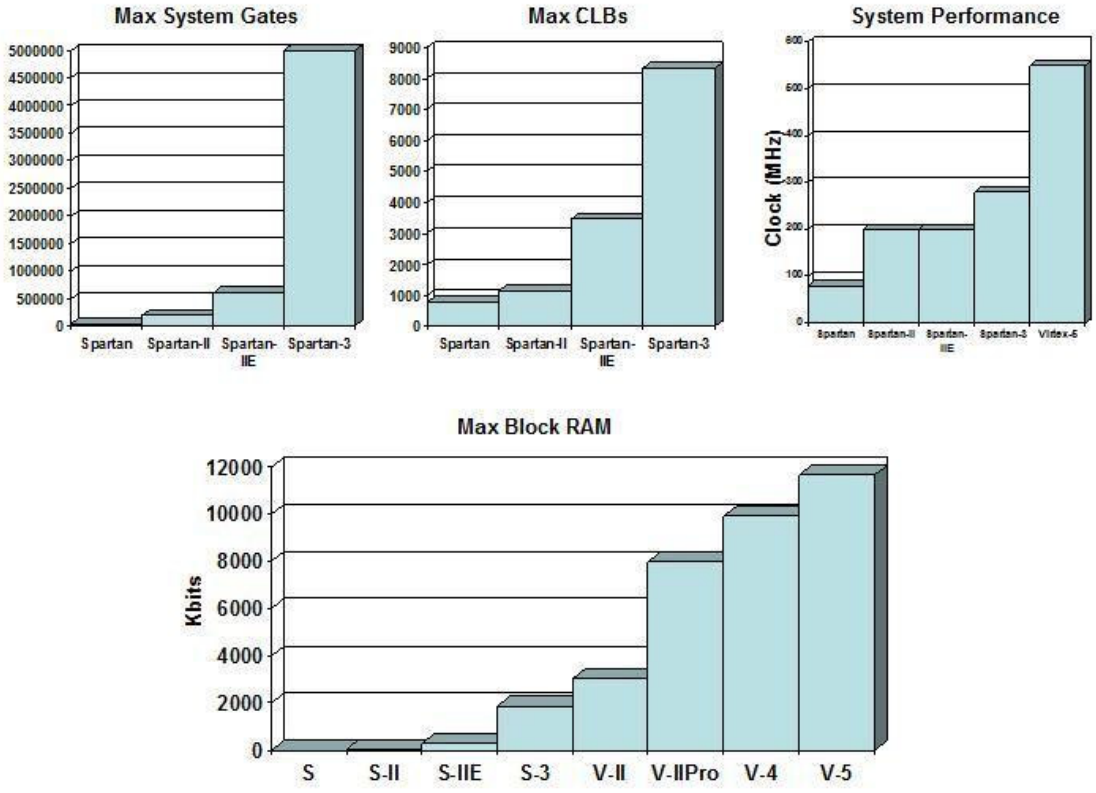

Figura 2.4 – Evolução das FPGAs

A FPGA é apresentada como um componente de grande qualidade e com alto desempenho. A sua evolução conduziu-a a uma densidade elevada e impôs baixas tensões de alimentação (ver Figura 2.4). O desenvolvimento do processo tecnológico torna possível colocar mais transístores num chip, que em chips com tecnologias mais antigas com o mesmo tamanho. Assim, no caso da implementação do mesmo circuito, um chip de menor tamanho é suficiente para conduzir a uma solução de menor custo. No entanto, um aumento da densidade e da dimensão da lógica pode causar aumento do consumo. Isto resulta numa maior dissipação de calor tornando-se um problema bastante grave. Naturalmente, é deveras importante baixar o consumo, e para isso é necessário reduzir a dissipação de calor (ver Figura 2.5). Há muitas maneiras de se reduzir o consumo. Uma das formas é baixar a tensão de alimentação (Figura 2.6) [9]. Além disso, ela está directamente relacionada com a potência. Por exemplo, se a fonte de alimentação de tensão for reduzida para metade, o consumo de energia será reduzido para um quarto. Para além disso, com a redução da tensão tornam-se mais fáceis as operações de alta velocidade. Devido ao descrito anteriormente, a elevada performance

na FPGA move-se no sentido de a tensão de alimentação baixar para níveis mais pequenos, causando contudo falhas entre as interfaces externas. Devido a este facto, a maioria das FPGAs tem mais de duas linhas de alimentação. Assim, as FPGAs podem ter a melhor performance internamente, já que funcionam a tensões mais reduzidas, e as partes que interligam a interfaces externas operam a tensões adaptadas às interfaces.

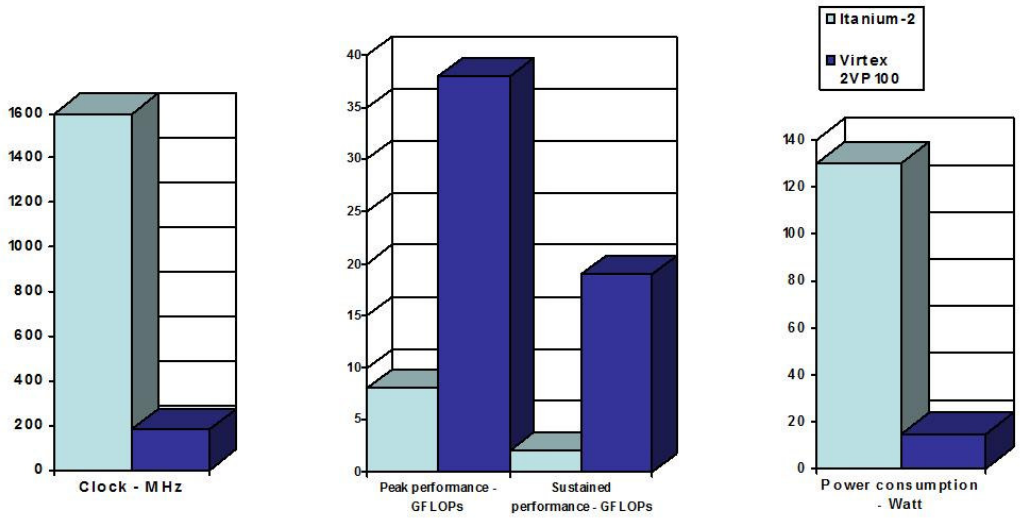

Figura 2.5 – Itanium-2 vs Virtex2 [8]

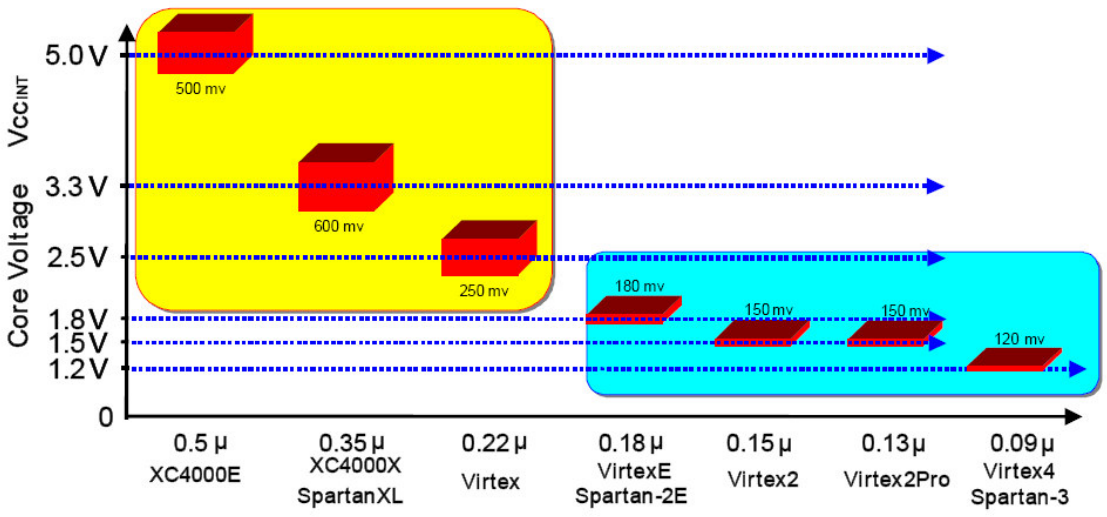

Figura 2.6 – Evolução das tensões nas FPGAs

A Figura 2.5 mostra três gráficos que permitem comparar uma FPGA (Virtex2) e o Itanium-2 que é uma família de processadores da Intel de 64 bits. Pode-se verificar que apesar de a FPGA utilizar uma frequência de relógio bastante mais baixa, esta consegue obter melhores performances e consumir menos energia.

2.4 Evolução das placas de prototipagem

As inúmeras vantagens das FPGAs, fazem com que as placas de prototipagem sejam o alvo ideal para sistemas embutidos modernos. Um grande número de placas de prototipagem, baseadas em FPGAs, tem vindo a ser fabricadas para permitir a verificação de soluções alternativas e competitivas em engenharia. O uso destas placas de prototipagem simplifica, significativamente, a implementação de novas aplicações para a FPGA e permite que o tempo de desenvolvimento seja reduzido.

Nos últimos dez anos, o departamento vem adquirindo e utilizando várias placas, quer para o processo educativo quer para investigação. Na Tabela 2.2 pode-se verificar as placas utilizadas e o ano da sua construção.

Tabela 2.2 – Placas de desenvolvimento utilizadas no DETI nos últimos 10 anos

Placa	**FPGA**	**Ano**
Annapolis FireFlyTM	XC6216/6264	1997 - 2001
XESS XS40	XC4010XL	1998 - 2002
Alpha Data ADM-XRC	Virtex-EM XCV812E	2000 - 2003
Alpha Data ADM-XPL	Virtex-II Pro XC2VP7	Desde 2003
XESS XSA100	Spartan-II XC2S100	Desde 2002
Trenz TE-XC2Se	Spartan-IIE XC2S300E/400E	Desde 2002
Celoxica RC100	Spartan-II XC2S200	Desde 2002
Celoxica RC200	Virtex-II XC2V1000	Desde 2003
Celoxica RC10	Spartan-3 XC3S1500L	Desde 2005

As placas referidas na Tabela 2.2, na sua maioria apresentam-se com vários dispositivos

periféricos e com poucos pinos de entrada/saída. O elevado número de interfaces e dispositivos periféricos, tornam uma placa de prototipagem mais dispendiosa. O número de utilizações desses mesmos periféricos, no processo educativo, é relativamente baixo, e a necessidade de ter várias placas faz com que o investimento seja elevado e o aproveitamento baixo. A existência de poucos pinos de entrada/saída disponíveis faz com que seja quase inexequível a integração de placas externas ou periféricos que a placa não contemple.

Outro aspecto importante tem a ver com o modo de programação da FPGA. As placas referidas na Tabela 2.2 são programadas no modo série, sendo mais lento que o modo paralelo.

Considerando os aspectos referidos anteriormente, optou-se por desenvolver a placa de prototipagem DETIUA-S3.

O objectivo inicial consistiu em desenvolver uma placa que integrasse um módulo USB, que permitisse a ligação a um computador para troca de dados e para a alimentação da mesma, que disponibilizasse o máximo de pinos de entrada/saída, permitisse guardar vários ficheiros de configuração (*bitstreams*), fosse rápida no modo de programação da FPGA e tivesse um baixo custo.

A possibilidade de desenvolver uma placa com as características pretendidas, permitiu, além de se obter a placa desejada, adquirir experiência ao nível do *hardware*, algo que ainda não tinha sido realizado com profundidade até então.

No mercado, existe um grande número de empresas que desenvolve placas de prototipagem baseadas em FPGAs, tal como se pode verificar em [10]. Nesta lista de placas é possível analisar a evolução existente das FPGAs utilizadas e ao mesmo tempo o progresso existente das placas dentro da mesma empresa.

2.5 Características importantes

Não é possível, de uma forma simples, dizer qual ou quais as características mais importantes de uma placa de desenvolvimento utilizando uma FPGA. Isso varia de caso para caso. No processo educativo, podemos afirmar que os recursos da FPGA, o manuseamento da placa, o preço e o número pinos de entradas e saídas são características muito importantes.

Tal como foi descrito acima, os recursos da FPGA são uma característica fundamental

numa placa de desenvolvimento. Esta tem sofrido um desenvolvimento muito grande ao longo do tempo, quer ao nível dos recursos quer ao nível da tecnologia por si utilizadas. Este desenvolvimento rápido faz com que as placas fiquem desactualizadas e portanto existe a necessidade de se adquirir placas mais recentes. As placas referidas na Tabela 2.2, tem como principal diferença as FPGAs utilizadas. Quanto aos periféricos disponíveis não existem grandes disparidades. Os periféricos mais usuais neste tipo de placas são a porta paralela, porta série, VGA, LCD, rato e teclado. Mais recentemente, as empresas que desenvolvem placas de prototipagem, têm optado por outro tipo de periféricos como por exemplo: porta USB, interface Ethernet, PCI e *Bluetooth*.

Uma outra característica importante passa pela existência do maior número de pinos de entrada e saída. Tal permite anexar outras placas ou periféricos. As placas de desenvolvimento na sua maioria têm dispositivos periféricos integrados, mas nem sempre os mais necessários, e quando estes dispositivos periféricos existem, há a tendência de se dispor de poucos ou nenhuns pinos de entrada/saída.

No que se refere ao processo educativo, mais concretamente a alunos que normalmente não têm qualquer experiência anterior nesta área, é necessário simplificar ao máximo o seu manuseamento de forma a evitar a perda de tempo em processos que não são importantes para avaliação. O manuseamento tem a ver com o processo de envio do ficheiro de configuração, armazenamento e acesso a dados, se a memória é volátil ou não, etc. Estas características de manuseamento, normalmente têm a ver com a interface desenvolvida para o efeito. Algumas das placas referidas na Tabela 2.2 não possuem uma interface muito amigável. As funções que, normalmente, essas interfaces possuem são o envio de ficheiros de configuração e de dados para uma dada zona de memória previamente definida.

2.6 Conclusões

Os dispositivos lógicos programáveis têm sofrido uma grande evolução, o que tem levado a um crescendo no interesse por parte dos utilizadores de sistemas digitais. Esse interesse, por sua vez, também obriga as empresas a evoluir, quer na sua tecnologia quer nas ferramentas de CAD associadas.

A evolução dos sistemas reconfiguráveis, descrita neste capítulo, permite entender de uma forma mais clara o aparecimento da placa DETIUA-S3.

As placas utilizadas pelo DETI, nos últimos anos, tinham um valor elevado e por isso era difícil adquirir, com alguma frequência, placas com FPGAs recentes e portanto sentiu-se a necessidade de criar uma placa mais acessível, de fácil utilização e adaptada às necessidades. A placa DETIUA-S3 consegue satisfazer os pontos pretendidos quer para o presente quer para o futuro [11].

Capítulo 3

Placa DETIUA-S3

Sumário

Para tentar dar resposta ao problema do custo, sem impor restrições de aplicabilidade, surgiu a necessidade de desenvolver uma placa de prototipagem baseada numa FPGA, contendo apenas dispositivos essenciais a uma fácil configuração da placa e barramentos de expansão.
Este capítulo descreve todo o desenvolvimento e concepção da placa de prototipagem DETIUA-S3. Trata-se de uma placa de baixo custo e que foi desenvolvida numa primeira fase para o processo educativo, criando assim uma mais valia no campo dos sistemas reconfiguráveis.

3.1 Introdução

O mercado encontra-se fértil no que se refere à oferta de placas de prototipagem. Acontece que nem sempre é fácil encontrar uma que vá de encontro aos interesses pretendidos. Ter uma placa que permite fazer aquilo que se pretende e ao mesmo tempo poder alterar determinados aspectos de forma a adaptar a placa às nossas necessidades, levou ao desenvolvimento da placa de prototipagem DETIUA-S3, que tem como componente central uma FPGA Spartan-3 da Xilinx.

3.2 Dispositivo Lógico Reprogramável – FPGA

O componente reconfigurável adoptado foi a FPGA Spartan-3 XC3S400 PQ208, da XILINX, baseada na tecnologia 90nm, comportando 400 mil portas de sistema, 56Kb de RAM distribuída, 288 Kb de blocos de RAM, 16 multiplicadores dedicados e

disponibiliza 137 pinos de entrada/saída [12].

As FPGAs da família Spartan-3 apresentam características de baixo custo, lógica de alta performance para grandes volumes, sendo úteis para aplicações orientadas ao consumo.

No momento em que se começou a desenvolver a placa, a FPGA a ser utilizada ainda não existia no mercado. Por isso, inicialmente, recorreu-se a *Engineering Samples* (ES) disponibilizadas pela Xilinx, o que permitiu, numa primeira fase, conhecer o seu funcionamento e desenvolver uma pequena placa protótipo para implementação de pequenos projectos. Nesta fase, a configuração da FPGA era realizada através do cabo JTAG e da ferramenta iMPACT da Xilinx. Esta ferramenta de *software* é utilizada para configuração e programação de todas as PLDs (FPGAs e CPLDs) e PROMs da Xilinx. Os pequenos projectos implementados permitiram, por exemplo, fazer o controlo dos blocos que posteriormente foram integrados na placa tal como a memória *flash* e o módulo USB.

3.3 Arquitectura da Placa

3.3.1 Com módulo USB

Tal como foi referido no ponto anterior, o dispositivo central escolhido foi uma FPGA. Por si só, não é autónoma o suficiente para que se possa considerar uma placa de prototipagem. Para isso, foi necessário introduzir alguns dispositivos, que têm como função prestar apoio à configuração da FPGA e ao bom funcionamento da placa.

O desenvolvimento desta placa começou pela compreensão e pela realização de testes da FPGA escolhida. Conhecido todo o funcionamento do componente central e conhecendo antecipadamente os objectivos finais da placa, integraram-se os dispositivos essenciais para uma fácil configuração e manuseamento.

Depois de todo o sistema analisado, obteve-se o diagrama de blocos indicado na Figura 3.1.

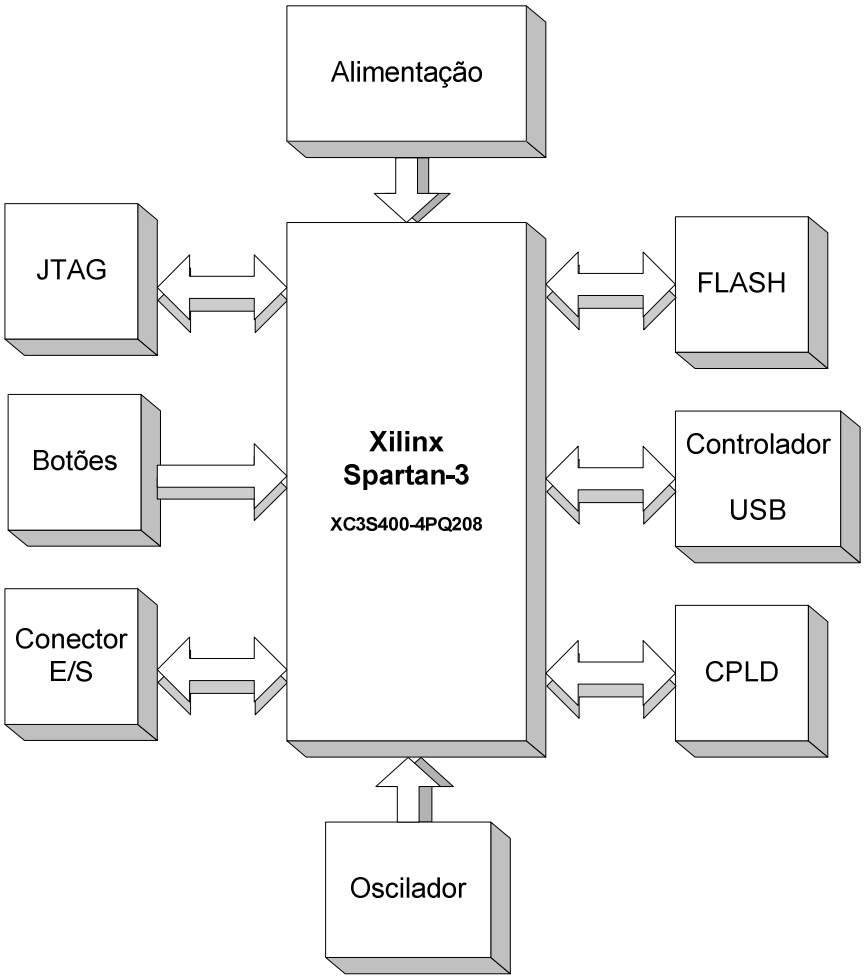

Figura 3.1 – Diagrama de blocos da placa DETIUA-S3

Um dos dispositivos incorporados foi o módulo DLP-USB245M da DLP Design [13], que é uma interface USB-paralelo (FIFO), compatível com USB 1.1 e USB 2.0. Ligando este módulo a um computador, é possível transferir dados a um máximo de 1 MB/s e, ao mesmo tempo, fazer uso da tensão de 5V fornecida pela porta USB à qual está ligada. Assim, um único cabo permite realizar toda a gestão e alimentação da placa, a partir de um computador. Este módulo pode ainda ser utilizado como interface para vários periféricos. O módulo DLP-USB245M tem uma interface simples, que permite um fácil controlo por parte da FPGA. Este módulo contém um FIFO de transmissão, um FIFO de recepção e uma EEPROM que permite armazenar palavras tais como o ID do módulo, entre outras. O protocolo é assegurado automaticamente pelo módulo, utilizando para o efeito um integrado da empresa FTDI e *drivers* disponibilizados pela mesma.

A placa inclui também uma CPLD, que é um dispositivo que integra várias PLDs

interligadas por uma estrutura programável. Tem como finalidade auxiliar o processo de configuração da FPGA, utilizando o barramento de endereços e pinos de controlo da memória *flash*. Internamente foi implementado uma máquina de estados finitos e um contador de endereços, na linguagem VHDL. De realçar, que se optou por disponibilizar um barramento de pinos de entrada/saída. A função deste barramento será descrita mais à frente.

Foi incorporada uma memória *flash* [14], que tem uma capacidade de 2 MB, com sectores de 64 Kb, exceptuando os últimos quatro. A memória foi dividida em três zonas lógicas tal como ilustra a Figura 3.2.

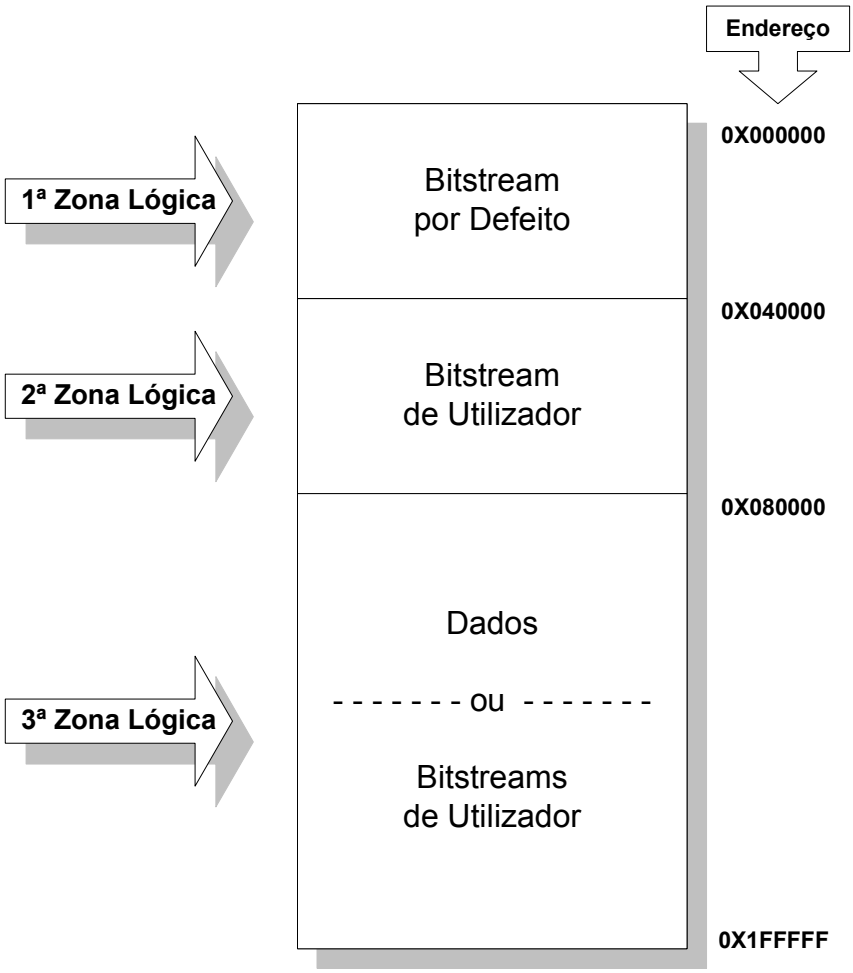

Figura 3.2 – Memória flash

A primeira zona lógica contém a *bitstream*, que permite a gestão de alguns dos elementos constituintes da placa, sendo também responsável por levar a cabo as operações solicitadas pelo PBM. Esta *bitstream* funciona como um *bootloader*. As *bitstreams* foram geradas pela ferramenta ISE da Xilinx, que permite desenvolver

projectos utilizando, neste caso, a linguagem VHDL

A segunda zona lógica tem como finalidade guardar a *bitstream* do projecto desenvolvido pelo utilizador. Esta *bitstream* é enviada através da aplicação Prototyping Board Manager (PBM) [15, 16] a correr no computador e armazenada na memória através da *bitstream* que está na primeira zona lógica, tendo previamente configurado a FPGA.

A terceira e última zona lógica, é destinada a um acesso livre por parte do utilizador, transferindo dados de/para o computador através do PBM. Esta zona também pode ser utilizada para guardar mais 6 ficheiros de configuração (*bitstream*). A selecção das *bitstreams* a utilizar é conseguida através de interruptores ligados ao barramento da CPLD, pressionando depois o botão "*Project*" para configurar a FPGA. Por outro lado, a activação de qualquer um destes *bitstreams*, para configurar a FPGA, pode ser despoletada pela própria FPGA, num contexto de reconfiguração.

Por se tratar de uma memória *flash*, as *bitstreams* e os dados armazenados não se perdem quando se interrompe a alimentação da placa. Deste modo, a placa pode ser utilizada autonomamente, isto é, sem recorrer a um dispositivo com porta USB, mas desde que alimentada por uma outra fonte, já que esta permite alimentação externa.

Numa primeira fase, o bloco de alimentação da placa foi concebida considerando integrados da Texas Instruments (TI) [17]. Foram utilizados 3 reguladores de tensão, para valores 1.2V, 2.5V e 3.3V que a FPGA precisa. Acontece que dois desses reguladores mostraram-se instáveis para o efeito e portanto foi necessário recorrer a outra solução. A TI no mesmo período que se estava a trabalhar neste bloco, lançou um novo integrado, TPS75003 (ver Figura 3.3), que foi concebido com o propósito de alimentar uma FPGA da família Spartan-3. O bloco de alimentação baseou-se então nesse integrado, que permite obter as três alimentações necessárias. Foi introduzido também um regulador de tensão de 5V que permite receber de uma fonte externa, que não a porta USB.

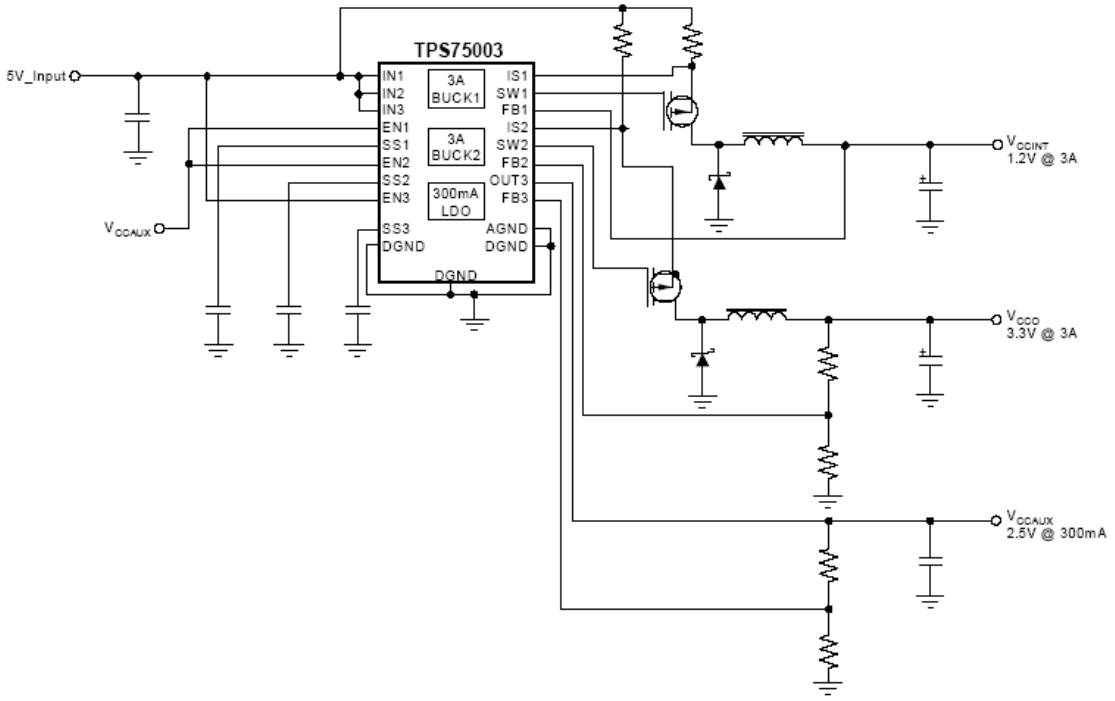

Figura 3.3 – Esquema geral da alimentação

Existem dois barramentos de expansão, um com 50 pinos de entrada/saída e outro com 30, que permitem a ligação de inúmeros dispositivos, tais como teclado, rato, monitor VGA, porta RS232, placas de expansão, etc.

Existe ainda um outro barramento que disponibiliza, além da massa (GND), as tensões 5V, 3.3V, 2.5V e 1.2V.

Na Figura 3.4 é possível verificar a arquitectura final da placa DETIUA-S3.

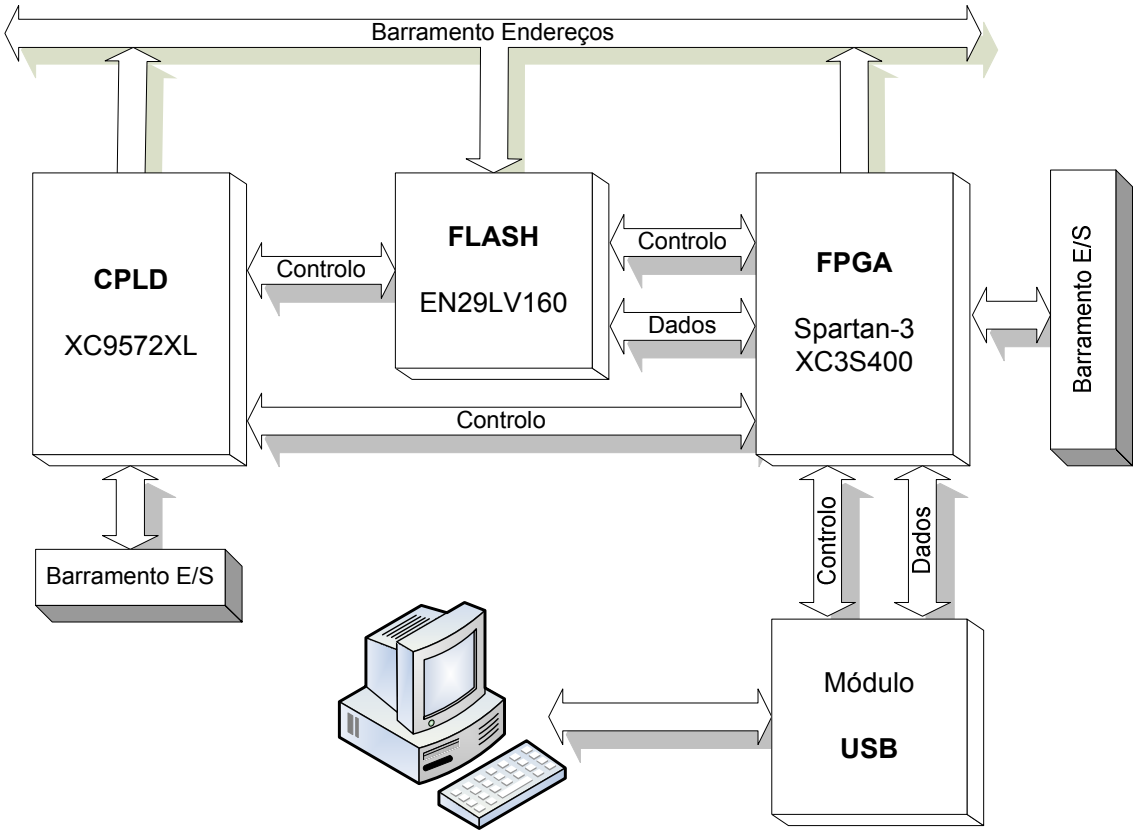

Figura 3.4 – Arquitectura da placa com módulo USB

3.3.2 Com módulo Bluetooth

Em relação à arquitectura descrita anteriormente, existe a alteração de um dos dispositivos. O módulo USB desaparece da arquitectura original, sendo substituído por um módulo *Bluetooth* (ver Figura 3.5).

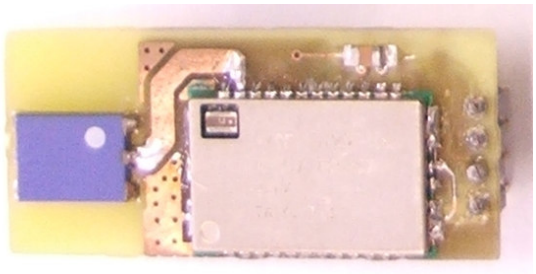

Figura 3.5 – Módulo *Bluetooth*

O módulo *Bluetooth* utilizado apresenta uma interface série, isto é, o dispositivo que faz o controlo do módulo vê uma porta série com um *Baud Rat*e de 115200, 8 bits de dados,

sem paridade e com 1 stop bit. O módulo está programado internamente, de fábrica, de modo a funcionar desta forma.

Da mesma maneira que o módulo USB, este módulo também permite ser utilizado como interface para outros dispositivos periféricos.

A placa original, isto é, com o módulo USB, permite que no mesmo *socket* que é encaixado o módulo USB, ser embutido o módulo *Bluetooth*. Obviamente, que as ligações não são as mesmas já que o módulo USB apresenta várias linhas de entrada/saída enquanto que o módulo Bluetooth apresenta apenas duas que são o RX e o TX, não entrando em conta com a alimentação.

A introdução deste módulo vai levar à alteração do tipo de controlo pela parte da FPGA, pelas razões especificadas anteriormente.

A arquitectura da placa utilizando o módulo Bluetooth é apresentada na Figura 3.6.

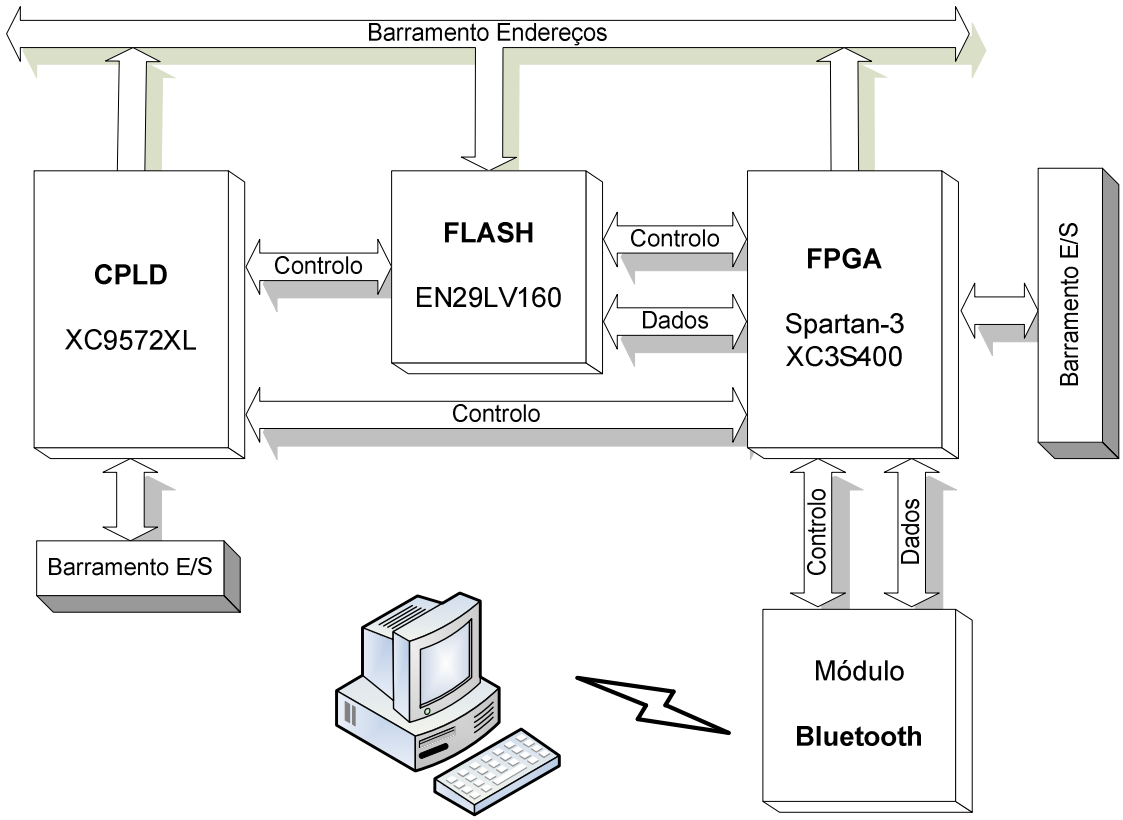

Figura 3.6 – Arquitectura da placa com módulo Bluetooth

3.4 Desenvolvimento

Tal como foi referido anteriormente, a forma final da placa foi conseguida através de uma parte de investigação, seguida de um conjunto de experiências que possibilitaram averiguar toda a teoria que estava por trás.

Depois de escolhidos os componentes necessários para a implementação da placa desejada, foi necessário testar todos esses dispositivos individualmente.

O primeiro dispositivo a ser testado foi a FPGA Spartan-3, já que se tratava do dispositivo nuclear do sistema. Escolheu-se um dispositivo da Xilinx, devido a esta empresa ser a melhor do mercado, e também de uma vasta experiência, que existia no departamento, nas FPGAs da mesma. A Xilinx tem duas famílias de FPGAs, que são a Virtex e a Spartan. Optou-se pela família de mais baixo custo, Spartan, mas dentro dessa família a que existia no mercado era a SpartanII-E. Contudo, a Xilinx estava a anunciar uma nova FPGA, Spartan-3. Esta FPGA apresentava vantagens em relação à anterior, mas devido a não ter sido ainda lançada no mercado, apenas podia ser escolhida se a Xilinx fornecesse *engineering samples* (ES). Depois de se ter obtido uma resposta favorável da empresa, optou-se então pela Spartan-3. Devido à falta de experiência a nível de *hardware* e de se tratar de uma FPGA recente, foi necessário desenvolver uma placa na qual apenas estava incluída a FPGA e os condensadores de desacoplamento. Após este procedimento, foram desenvolvidos vários testes, que numa primeira fase incidiram fundamentalmente no modo de configuração. Era necessário compreender a função de todos os pinos de configuração e só a partir daí, escolher o modo mais útil ao nosso sistema.

O dispositivo seguinte a ser testado foi a memória flash. Desenvolveu-se uma placa com a memória e com a placa com a FPGA desenvolvida anteriormente fez-se todo o controlo. Com este teste chegou-se à conclusão, que havia necessidade de introduzir um elemento com o objectivo de controlar apenas a memória. Optou-se por uma CPLD, que permitia, inclusivamente, utilizar grande parte do código VHDL desenvolvido nos testes com a FPGA.

Com estes três dispositivos foi possível testar o modo de configuração da FPGA. O modo escolhido foi o paralelo, já que permite uma velocidade maior do que o modo série. Este teste apresentou algumas dificuldades devido a Xilinx não ter disponível, de uma forma clara, toda a informação necessária.

Faltava apenas um dos dispositivos necessários ao bom funcionamento da placa. Esse dispositivo era um módulo USB, que tem uma interface USB-paralelo. Existiram algumas complicações iniciais, já que foi necessário desenvolver uma aplicação para correr num computador e que permitia comunicar com o módulo USB. Os *drivers* fornecidos pela empresa do módulo também permitiram emular uma porta USB como se fosse uma porta série e numa primeira fase, podia-se utilizar qualquer programa, que existe no mercado, que comunicasse com a porta série de um computador. Obviamente, foi desenvolvido uma aplicação com base nos *drivers* fornecidos e que irá ser descrita mais à frente.

Depois de testados, todos os dispositivos, foi necessário implementar a parte da alimentação da placa. Esta parte tornou-se extremamente complexa. A alimentação externa era realizada através da porta USB e portanto tinha-se 5V disponíveis. Devido à limitação de corrente da porta USB, 500mA, foi necessário quantificar a corrente total que iria ser necessária. Realizados alguns testes, verificou-se que o valor ficava abaixo do limite da porta USB. A FPGA necessitava de 3 tensões diferentes, 3.3V V_{CCO}, 2.5V V_{CCAUX} e 1.2 C_{CCINT}. Depois de alguma pesquisa encontraram-se 3 reguladores de tensão que, à partida, encaixavam nos parâmetros pretendidos [18]. Os testes revelaram que a escolha não tinha sido a melhor e portanto havia a necessidade de substituir os que não funcionavam bem. Depois de uma nova pesquisa, encontrou-se um dispositivo criado, pela Texas Instrument (TI), que tinha como objectivo fornecer as tensões de alimentação para a família de FPGAs Spartan-3. Como se pode verificar pela Figura 3.3, as correntes de saída possíveis dos vários ramos vão muito além da corrente limite da porta USB. Claro que essas correntes só são atingidas, se nos respectivos ramos for colocado uma carga que o obrigue. Portanto, ao circuito apresentado pela TI, foram feitas algumas alterações aos valores dos componentes de maneira a baixar essas correntes máximas possíveis de obter.

Obtida a arquitectura, o passo seguinte consistia na criação da própria placa.

A placa DETIUA-S3 foi desenhada na ferramenta OrCAD [19]. Primeiramente, foi criado o esquemático no OrCAD Capture. Nesta fase são colocados os dispositivos que a placa contém e todas as ligações existentes entre esses mesmos dispositivos. A ferramenta tem um conjunto de livrarias que contêm vários dispositivos, que por vezes não existem, e é necessário criar ou adaptar um já existente às reais necessidades.

Com o esquemático criado e utilizando o OrCAD Layout, decorreu o processo de criação da placa final. Aqui, o processo é algo moroso devido à necessidade de se ter que encontrar nas livrarias o componente respectivo, por outras palavras, é necessário seleccionar o *package* indicado para esse componente. Quando não existem os componentes nas livrarias da ferramenta então torna-se necessário criá-los.

A placa DETIUA-S3 foi desenvolvida com duas camadas. Na Figura 3.7 e Figura 3.8 pode-se observar a camada de cima e de baixo, respectivamente. Na camada superior, estão colocados os dispositivos principais, tais como o módulo USB/Bluetooth, oscilador, CPLD, FPGA, memória *flash*, barramentos e o bloco da alimentação.

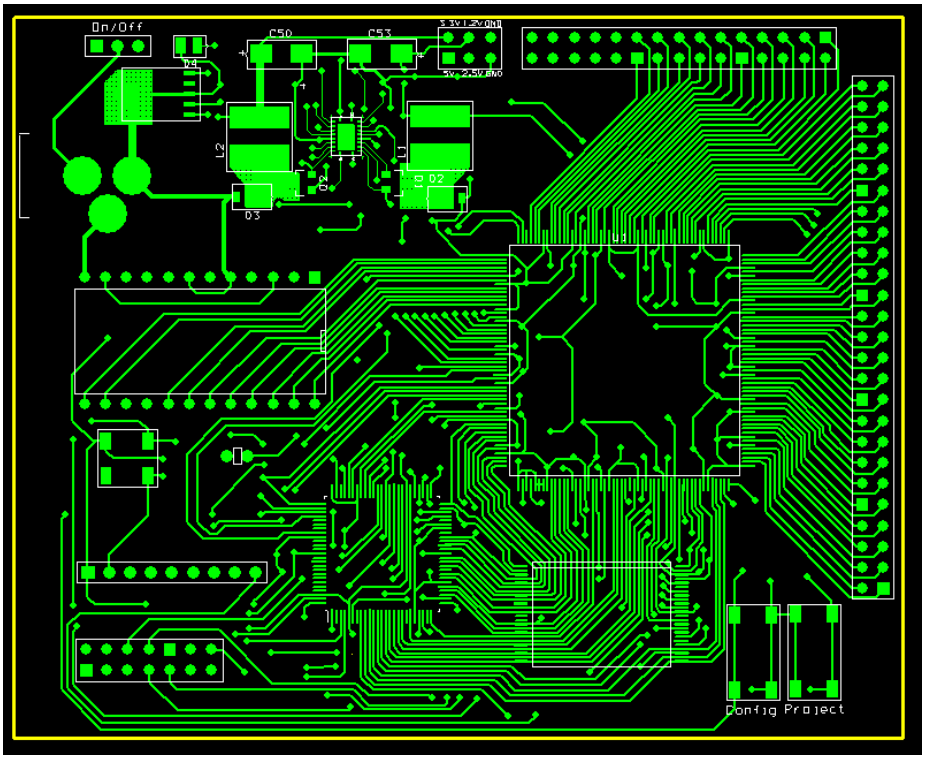

Figura 3.7 – PCB camada de cima com a disposição dos componentes.

Na camada inferior foram introduzidos os componentes de menor dimensão, tais como condensadores de desacoplamento, resistências de *pull-up* e de *pull-down*.

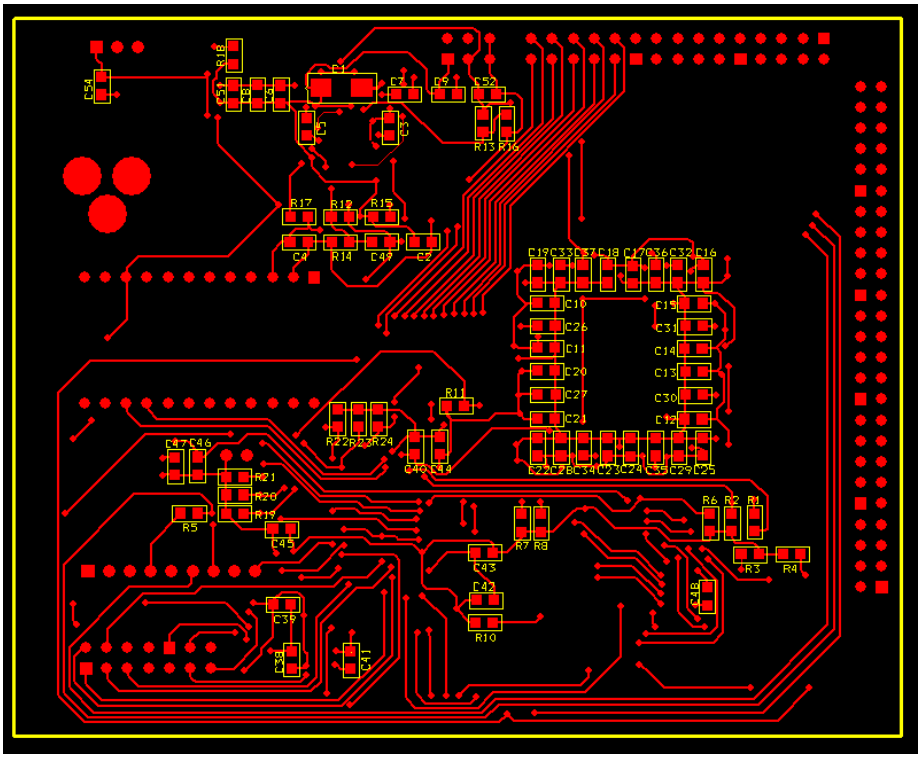

Figura 3.8 – PCB camada de baixo com a disposição dos componentes.

Após este processo de desenvolvimento ao nível do CAD, foram fabricadas as placas de circuito impresso, passando de seguida ao processo de soldadura dos componentes constituintes da placa DETIUA-S3.

De realçar que, antes desta versão, existiu uma outra que apresentou algumas anomalias, principalmente na parte da alimentação. Depois de corrigidas essas mesmas anomalias passou-se então à versão final.

Concluída esta fase, veio o processo de testes eléctricos e configuração. É necessário configurar dois componentes: um é a CPLD e o outro a FPGA. A configuração é realizada através do cabo JTAG da Xilinx. É necessário configurar uma vez a FPGA para possibilitar o envio, através da aplicação PBM, do *bitstream* por defeito para a primeira zona lógica da memória *flash*.

Através da Figura 3.9 e Figura 3.10 é possível observar a placa DETIUA-S3.

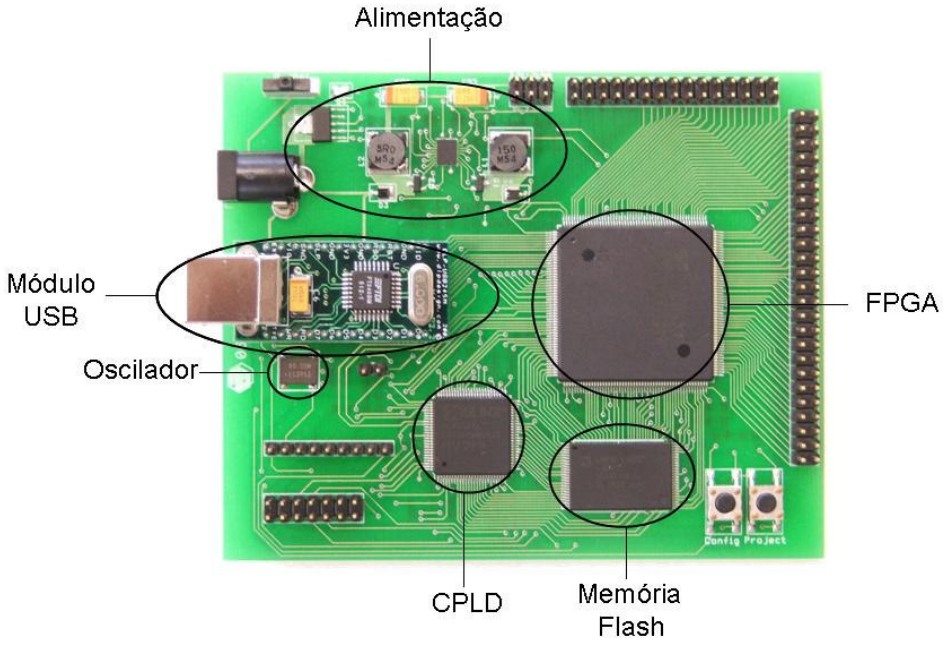

Figura 3.9 – Placa DETIUA-S3 com indicação dos módulos principais

Através da Figura 3.9 é possível verificar a localização dos módulos principais da placa. Na Figura 3.10 é possível ver os conectores incluídos na placa.

É de realçar que todos os pinos da FPGA estão a ser utilizados, isto é, estão ligados a dispositivos ou então disponíveis nos barramentos.

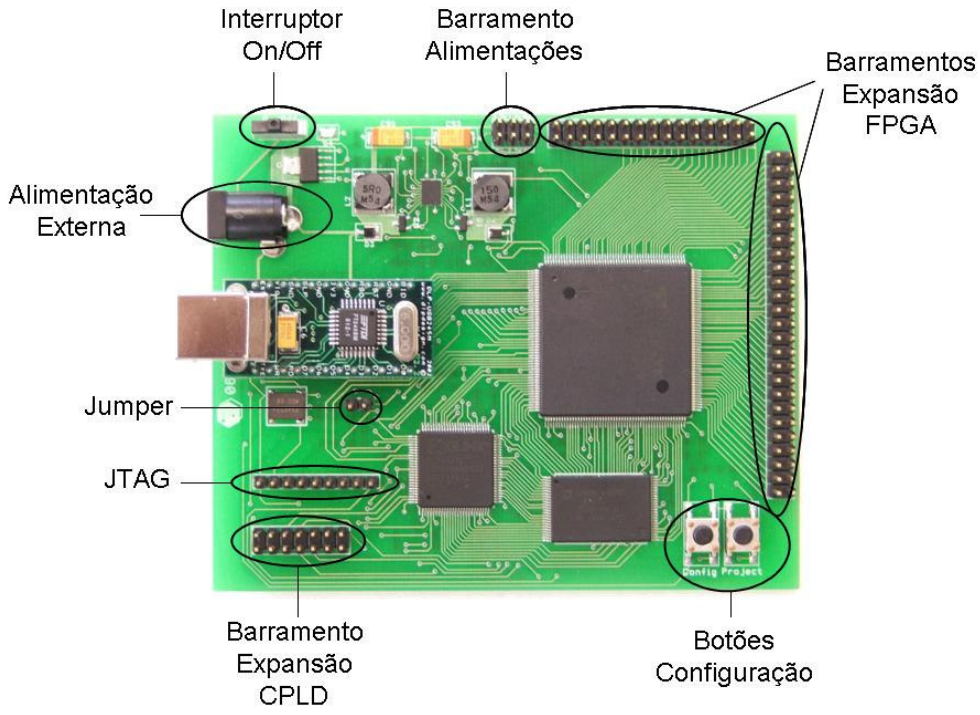

Figura 3.10 – Placa DETIUA-S3 com indicação dos conectores

3.5 Conclusões

Neste capítulo foi descrito todo o processo de desenvolvimento de uma placa de prototipagem. Foram apresentados os dispositivos constituintes da placa, e a interligação dos mesmos.

Para chegar a esta versão final da placa, muitas experiências foram realizadas levando, em alguns casos, a alterar algumas das características predefinidas. É de salientar que não existia muita experiência ao nível do *hardware* e portanto vários obstáculos apareceram ao longo deste trabalho. Alguns dos problemas passaram pela escolha das características pretendidas para a placa, a escolha dos componentes que a iria integrar, prospecção de mercado de produtos já existentes, escolha das ferramentas de desenvolvimento, etc.

Foram criados dois protótipos da placa até chegar à versão final. Esses protótipos acabaram por revelar alguns problemas ao nível do funcionamento de alguns componentes escolhidos, mais concretamente do sistema de alimentação.

A placa DETIUA-S3 tem duas versões, uma com módulo USB e outra com módulo *Bluetooth*, o que na altura constituiu por si só uma vantagem comparando com outras existentes no mercado, devido ao facto de poucas terem porta USB disponível ou *Bluetooth* incorporado.

Capítulo 4

Métodos e Ferramentas para Reconfiguração da FPGA

Sumário

O capítulo anterior faz uma exposição da placa de prototipagem desenvolvida. O processo seguinte, resume-se à configuração das PLDs (CPLD e FPGA) de modo a que se torne autónoma quanto à sua funcionalidade. As PLDs interagem entre si e fazem o controlo dos restantes dispositivos. Neste capítulo é descrito todo o processo de configuração da placa e ferramenta de auxílio PBM.

4.1 Introdução

As PLDs são dispositivos que necessitam de ser configuradas para fazerem uma dada tarefa. Após serem reconfigurados, através da porta JTAG, eles ficam aptos a executar a tarefa pré estabelecida e, em conjunto, funcionam como um sistema único que fica apto a receber "ordens" da ferramenta de gestão.

4.2 Hardware

4.2.1 Configuração Através da Porta USB

A comunicação entre a aplicação PBM e a placa DETIUA-S3 pode ser efectuada através da porta USB. A aplicação PBM utiliza os *drivers* fornecidos pela empresa FTDI que desenvolveu o módulo e usa um protocolo (ver Figura 4.1) criado para o efeito.

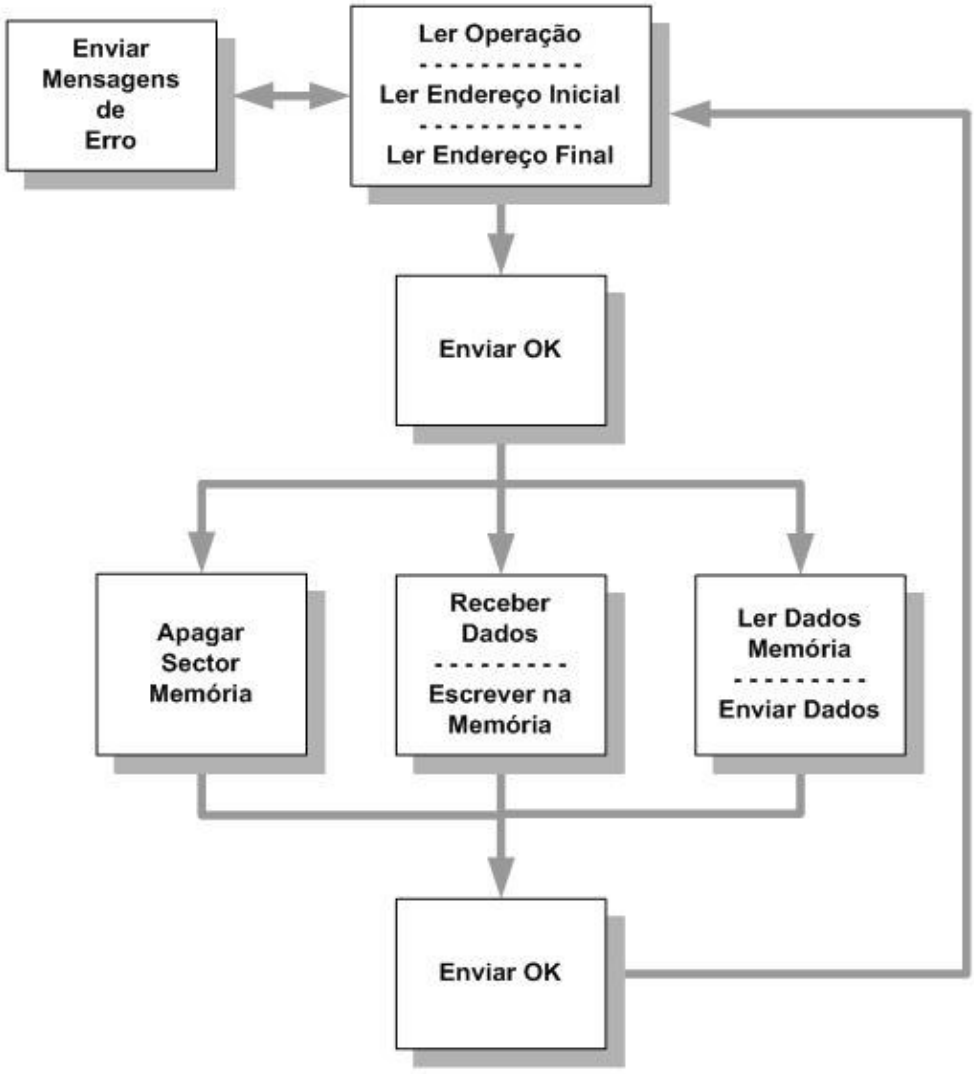

Figura 4.1 – Protocolo de comunicação

O protocolo permite todo o controlo, pela aplicação PBM, da memória *flash* utilizada na placa. O *bitstream* que se encontra armazenado na primeira zona lógica e que foi desenvolvido em VHDL contém este protocolo.

Na Tabela 4.1 encontram-se os comandos necessários para efectuar o controlo da memória flash. É de salientar que sempre que se efectua uma escrita numa zona da memória, é necessário apagar essa zona previamente.

Pode-se verificar através da Figura 4.1 que o protocolo implementado na FPGA permite apenas executar as várias operações aí referidas. Portanto, quem tem a função de gerir todo o controlo da memória *flash* é a aplicação PBM. Isto é, se a aplicação pretender enviar um ficheiro para a memória, ela tem que enviar a operação de apagar um sector

ou os sectores que esse ficheiro vai ocupar, e só depois é que envia a operação de escrita.

Tabela 4.1 – Comandos para controlo da memória flash

Command Sequence (Note 1)			Cycles	Bus Cycles (Notes 2–5)											
				First		Second		Third		Fourth		Fifth		Sixth	
				Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data
Read (Note 6)			1	RA	RD										
Reset (Note 7)			1	XXX	F0										
Autoselect (Note 8)	Manufacturer ID	Word	4	555	AA	2AA	55	555	90	X00	01				
		Byte		AAA		555		AAA							
	Device ID, Top Boot Block	Word	4	555	AA	2AA	55	555	90	X01	22C4				
		Byte		AAA		555		AAA		X02	C4				
	Device ID, Bottom Boot Block	Word	4	555	AA	2AA	55	555	90	X01	2249				
		Byte		AAA		555		AAA		X02	49				
	Sector Protect Verify (Note 9)	Word	4	555	AA	2AA	55	555	90	(SA) X02	XX00				
											XX01				
		Byte		AAA		555		AAA		(SA) X04	00				
											01				
CFI Query (Note 10)		Word	1	55	98										
		Byte		AA											
Program		Word	4	555	AA	2AA	55	555	A0	PA	PD				
		Byte		AAA		555		AAA							
Unlock Bypass		Word	3	555	AA	2AA	55	555	20						
		Byte		AAA		555		AAA							
Unlock Bypass Program (Note 11)			2	XXX	A0	PA	PD								
Unlock Bypass Reset (Note 12)			2	XXX	90	XXX	00								
Chip Erase		Word	6	555	AA	2AA	55	555	80	555	AA	2AA	55	555	10
		Byte		AAA		555		AAA		AAA		555		AAA	
Sector Erase		Word	6	555	AA	2AA	55	555	80	555	AA	2AA	55	SA	30
		Byte		AAA		555		AAA		AAA		555			
Erase Suspend (Note 13)			1	XXX	B0										
Erase Resume (Note 14)			1	XXX	30										

O protocolo da Figura 4.1 contém alguns dos comandos permitidos pela memória flash. O protocolo foi implementado através de uma máquina de estados finitos (MEF), desenvolvida em VHDL, envolvendo uma grande complexidade. A MEF, além de controlar a memória, também se encontra interligada com a CPLD e controla o módulo USB. O barramento de endereços da memória é partilhado com a FPGA e a CPLD, havendo a necessidade de os dois dispositivos estarem sincronizados de modo a evitar um acesso simultâneo à memória.

A configuração da FPGA através da porta USB é inicializada na aplicação PBM, que permite enviar o *bitstream*, do projecto, que foi previamente desenvolvido. A FPGA que anteriormente foi configurada com o *bitstream*, que está armazenado na primeira zona da memória *flash*, vai receber o *bitstream* do projecto enviado pelo PBM e armazena-o na segunda zona lógica. Armazenado, o utilizador tem que pressionar o botão de pressão *Project* que existe na placa, de modo que a FPGA se configure. De salientar que os dados são lidos pela FPGA através do módulo USB, byte a byte, dado tratar-se de um módulo USB-paralelo.

4.2.2 Configuração através do módulo Bluetooth

A placa DETIUA-S3 permite a integração de um módulo *Bluetooth*, que substitui o módulo USB. Basicamente, o módulo USB é removido do *socket* e é introduzido o módulo *Bluetooth* [20].

A configuração da FPGA utilizando o módulo *Bluetooth* e o módulo USB difere apenas no modo de funcionamento de cada. Devido ao funcionamento distinto de ambos os dispositivos, é inevitável que o controlo deles, por parte da FPGA, seja diferente, já que no módulo *Bluetooth* os dados são recebidos pela FPGA no modo série e do módulo USB no modo paralelo.

Na Figura 4.2 é apresentado parte do código, em VHDL, que faz a leitura dos dados enviados por um dispositivo *Bluetooth*. Este código permite ler uma trama com as características especificadas acima. Pode-se verificar na Figura 4.2 que a amostragem do bit é efectuada a meio do mesmo, garantindo assim uma maior precisão. Evidentemente que o número de amostragens poderia ser maior, conseguindo assim uma precisão ainda mais elevada. Mas para o efeito não foi considerado devido a se ter obtido bons resultados e portanto, com uma amostragem, estavam reunidas as condições necessárias.

O protocolo de comunicação utilizado é o mesmo que se apresenta na Figura 4.1.

```
-- Espera por um Start Bit
if  rx_process = '0' then

   if RX = '0' then
      rx_counter <=  "0101011000";--Amostra no centro da largura do bit
      rx_process <= '1';
      rx_bit_n <= 0;

   end if;
else
   if rx_counter = 0 then

      rx_counter <= "1010110000";

      case rx_bit_n is

         -- Confirma o start bit, caso contrario descarta os dados

         when 0 =>
                  if RX = '0' then
                    rx_bit_n <= rx_bit_n + 1;
                  else
                    rx_process <= '0';
                  end if;
         -- Data bits
         when 1|2|3|4|5|6|7|8 =>

                  rx_reg <= RX & rx_reg(7 downto 1);
                  rx_bit_n <= rx_bit_n + 1;

         --  Stop Bit
         when 9 =>
                  rx_process <= '0';
                  rx_bit_n   <= 0;
                  val_read   <= rx_reg;

      end case;
   else
      rx_counter <= rx_counter - 1;
   end if;
```

Figura 4.2 – Leitura dos dados através do módulo Bluetooth

Não obstante de o módulo *Bluetooth* apresentar uma taxa de transferência de dados mais baixa, esta via apresenta muitas vantagens tais como:

- Mobilidade, isto é, permite ter acesso, por parte de um utilizador, à placa em qualquer lugar;
- Flexibilidade, a tecnologia sem fio possibilita que a placa chegue onde os cabos não podem ir;
- É uma solução viável e de baixo custo para redes de curto alcance;
- É cada vez maior a quantidade de dispositivos com chips *Bluetooth*.

4.3 Software

4.3.1 Ferramenta de gestão à placa DETIUA-S3

Para facilmente se tirar o máximo partido das potencialidades da placa DETIUA-S3, foi desenvolvida, com o Bruno Pimentel, a aplicação Prototyping Board Manager (PBM) [15, 21]. Esta aplicação apresenta uma interface simples e amigável (ver Figura 4.3) e vem acompanhada de um manual de utilizador que está disponível através dos menus da mesma. Ela permite, entre outras funcionalidades úteis, enviar *bitstreams* para configurar a FPGA, transferir dados de um computador para a memória *flash* da placa e vice-versa e apagar sectores dessa mesma memória. A Figura 4.4 apresenta a árvore de menus de acesso às funcionalidades disponibilizadas por esta aplicação.

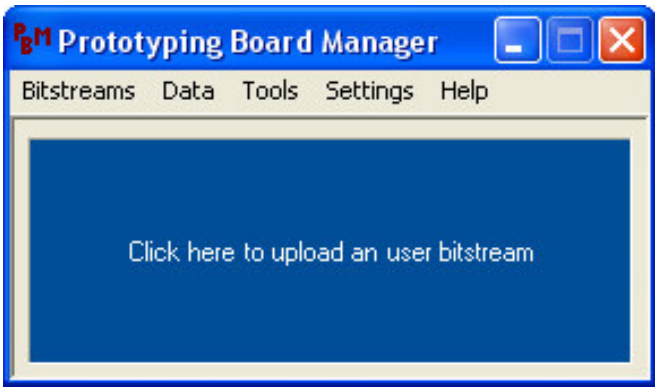

Figura 4.3 – Janela principal do PBM

Antes de o utilizador poder fazer uso destas funcionalidades, é necessário pressionar o botão "*Config*" para que o *bitstream* gravado na primeira zona lógica da memória *flash* seja carregado para a FPGA. As funcionalidades disponibilizadas pela aplicação são construídas a partir das operações básicas que o protocolo suporta, e que passa por apagar um sector, ler de um intervalo de endereços e escrever uma sequência de *bytes*.

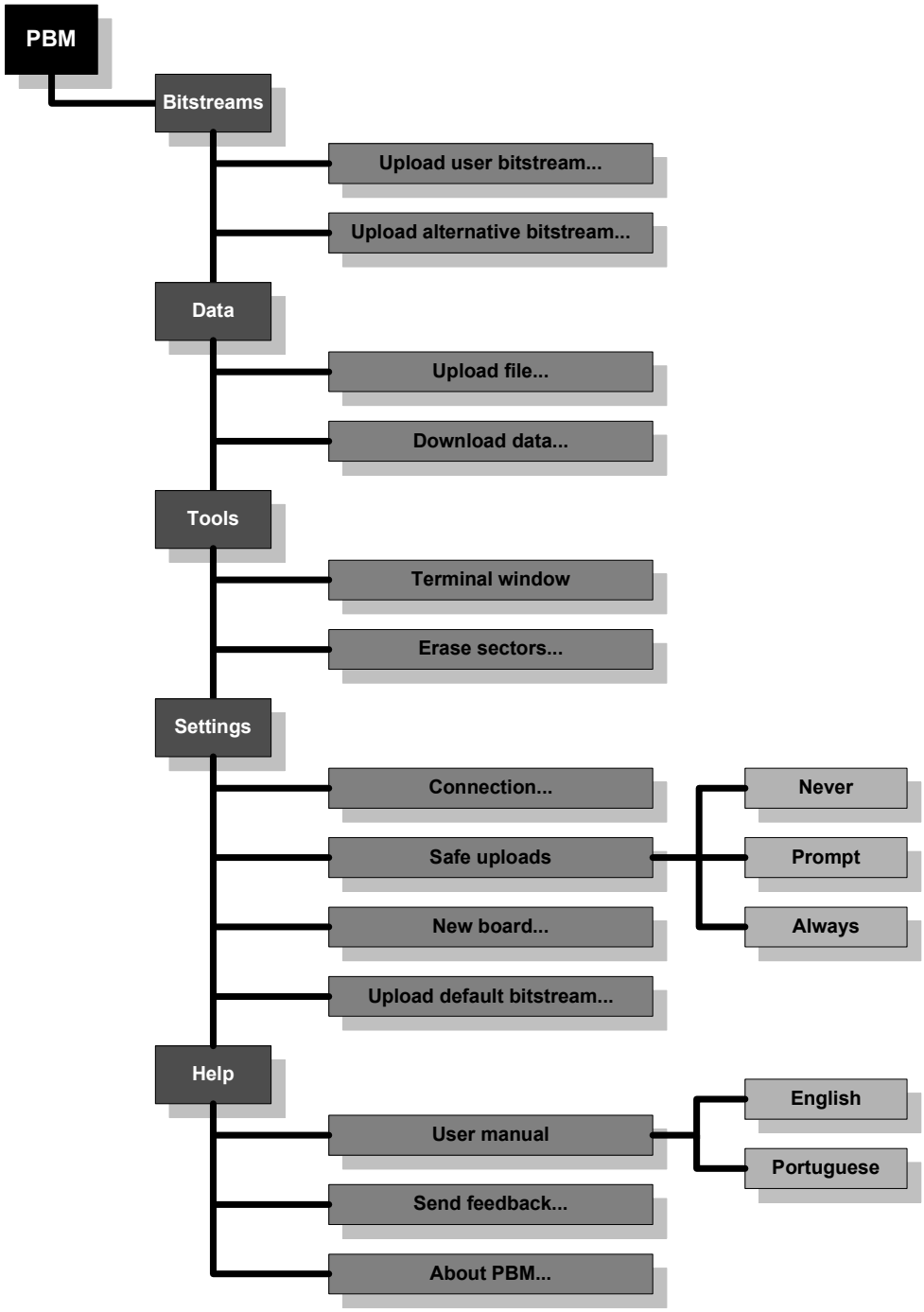

Figura 4.4 – Árvore de menus da aplicação PBM

A Tabela 4.2 apresenta os tempos médios necessários para a realização de algumas tarefas de maior utilidade. Note-se que cada sector tem 64 KB; que os tempos de escrita incluem o tempo de apagar os sectores envolvidos; e que a escrita de uma *bitstream* para a FPGA em questão envolve 4 sectores.

Tabela 4.2 – Tempo médio necessário para a execução de algumas tarefas

	Tempo Médio	
Tarefa	**USB**	**Bluetooth**
Apagar Sector	0,7''	1' 00''
Ler Sector	0,4''	1' 11''
Escrever Sector	1,5''	28''
Escrever Bitstream	5,5''	1' 27''

O PBM inclui ainda uma janela de terminal para a troca de dados entre o utilizador e a placa de prototipagem, em tempo de execução, através das ligações USB e *Bluetooth*. Esta ferramenta (ver Figura 4.5) constitui, assim, um periférico de entrada e saída integrado, ideal para monitorizar, testar e integrar projectos em fase de desenvolvimento.

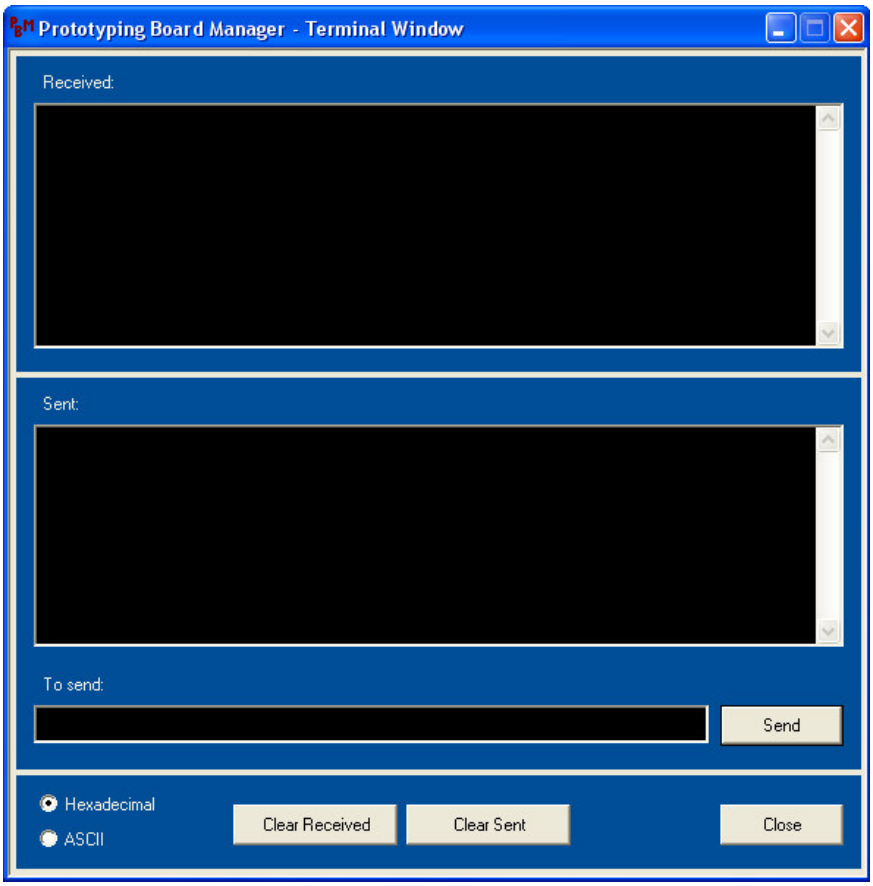

Figura 4.5 – Janela de terminal

4.4 Exemplo de configuração e interacção

O processo da configuração da FPGA principia na aplicação PBM. Esta permite o envio do ficheiro de configuração via USB ou *Bluetooth*, que será recebido pela FPGA que previamente foi configurada com o ficheiro de configuração por defeito. A FPGA recebe o ficheiro e armazena-o na memória *flash*. Posteriormente, é necessário pressionar o botão de pressão *Project* que faculta o processo de configuração da FPGA. Neste processo, a CPLD tem um papel fundamental que consiste no controlo da memória *flash*. De realçar que a configuração da FPGA é realizada no modo paralelo, permitindo uma configuração mais rápida da FPGA.

A configuração da FPGA pode ser realizada através do processo de pressionar o botão *Config* ou *Project*, tal como foi mencionado atrás, ou através de um pedido, por parte do programa a correr na FPGA, à CPLD. O pedido à CPLD consiste em enviar o endereço da zona de memória, onde se encontra o *bitstream* do projecto que se quer executar e autorizar a CPLD a enviar à FPGA um sinal de forma a que ela entre em modo de configuração, controlando de seguida a memória *flash*, de maneira que os dados sejam enviados para a FPGA. Se este último projecto, tal como todos os projectos que foram guardados na memória, incluir a opção de poder escolher qual o *bitstream* a configurar a FPGA, então o sistema encontra-se num processo de reconfiguração "dinâmica" da placa.

A interacção da placa de prototipagem e a aplicação PBM é realizada, tal como foi referido anteriormente, quer através de um módulo USB quer por um módulo *Bluetooth*. A escolha de um destes módulos influencia o funcionamento do sistema no que se refere aos tempos de transmissão de dados, sendo que no restante se torna inalterável o seu funcionamento. O protocolo existente entre um computador e a placa é o mesmo independentemente do módulo utilizado, não existindo qualquer limitação de funcionalidades por parte da placa devido à escolha de um dos módulos.

4.5 Conclusões

Com o trabalho descrito nos últimos dois capítulos, obteve-se um sistema reconfigurável, que permite implementar vários circuitos digitais.

A possibilidade de enviar o ficheiro de configuração para a placa DETIUA-S3 através

do módulo USB ou *Bluethoot* permite a integração da placa num vasto número de sistemas, que estão descritos nos capítulos seguintes.

É possível configurar uma FPGA de várias maneiras, tais como porta JTAG, modo série e modo paralelo. O método escolhido para a placa DETIUA-S3, modo paralelo, é o mais rápido.

Os fabricantes de FPGAs por norma disponibilizam os seus próprios programas de configuração ou então fazem parcerias com empresas que desenvolvem os mesmos, já que a tarefa de programar a FPGA não é trivial. Acontece que a ideia pretendida para a placa passava por ter uma aplicação que permitisse a troca de dados entre computador e FPGA. A utilização de um programa de configuração do fabricante da FPGA levaria a que o utilizador tivesse a necessidade de utilizar dois programas para utilizar a placa DETIUA-S3. A ferramenta PBM possibilita todo o tipo de manipulação de dados, desde a configuração da FPGA à troca de dados entre computador e FPGA.

Capítulo 5

Máquinas de Estados Finitos Reconfiguráveis Remotamente

Sumário

Neste capítulo será dado ênfase a um método para síntese de Máquinas de Estado Finitos (MEF) baseado num modelo de Hardware Template (HT). MEF é uma modelagem de um comportamento, constituído por um número finito de estados de memória que reagem ao mesmo sinal de relógio que é comum a todos os estados. HT é um circuito com uma estrutura predefinida que já foi implementado em hardware. Essa estrutura foi implementada através de memórias RAM e de *multiplexers*. Existem memórias que armazenam dados para controlo e outras que contém informação, isto é, os valores de saída. Por reprogramação dos blocos RAM, internos à FPGA, pode-se implementar uma MEF com funcionalidades distintas. É de realçar que o circuito da MEF resultante é muito rápido e qualquer transição de estado é realizado num ciclo de relógio.

Na parte final do capítulo será possível ver algumas aplicações práticas usando MEF reconfiguráveis.

5.1 Introdução

As máquinas de estado finito (MEF) [22, 23] são provavelmente os componentes mais usados em sistemas digitais. Isso, porque quase todas as ferramentas de desenho disponíveis, que são incluídas em sistemas de *Computer Aided Design* (CAD) industriais, permitem MEF sintetizadas a partir de especificações formais, tais como diagramas de estado, descrições de HDL, etc. A expansão rápida do mercado de

dispositivos de lógica programáveis (PLDs) como as FPGAs, incentivaram o desenvolvimento de sistemas de CAD tendo como alvo os PLDs, que incluem também uma variedade de ferramentas para síntese de MEF. A finalidade principal de um PLD é permitir que o mesmo *microchip* seja usado para construção de uma grande variedade de circuitos para diferentes aplicações práticas.

Para muitas dessas aplicações práticas, é desejável fornecer um circuito com potencialidades genéricas, mas possíveis de alterar. Os objectivos destes são muito diferentes. Para um dado tipo de aplicação pode-se pretender fornecer potencialidades similares às fornecidas para os microprocessadores de uso geral, que suportam o mecanismo de memória virtual. Em particular, isto permite que um dispositivo seja construído num *microchip*, que não tenha recursos de *hardware* suficientes para acomodar toda a funcionalidade do dispositivo. Para outro tipo de aplicações, pode ser desejável alterar o comportamento dependendo de eventos externos que não podem ser previamente conhecidos. Em alguns casos, é necessário fornecer flexibilidade suficiente para permitir alterações durante a etapa de depuração, etc. Estas exigências obrigam à procura de modelos de circuitos digitais que permitem alterações estáticas e dinâmicas na funcionalidade, inicialmente especificada. É de realçar, que enquanto os PLDs puderem ser programados e reprogramados durante a etapa de desenho, é usualmente difícil fazer alterações independentes do local após o processo de desenho ser finalizado. Por outras palavras, tais modificações requerem que a fase de desenho seja repetida, e esta pode introduzir novos problemas. Para qualquer especificação da MEF pode-se construir implementações em *hardware* e *software*. A implementação em *hardware* é geralmente muito rápida mas inflexível, significa que num momento de alteração na especificação inicial, todas as etapas de síntese da MEF devem ser repetidas desde do início. Uma implementação em *software* pode ser modificada facilmente, mas a MEF resultante é relativamente lenta e isso pode ser um problema para algumas aplicações práticas. A aproximação proposta à síntese de MEF é similar ao desenvolvimento em *software*. Decididamente, o recarregamento de programas na memória pode facilmente modificar a funcionalidade de um sistema em *software*. Vai ser demonstrado que a mesma aproximação, baseada em *hardware templates* (HT) reutilizáveis, pode ser utilizada em MEFs. Os componentes básicos do circuito podem ser reprogramados de tal forma que é possível implementar diferentes funcionalidades.

Assim a alteração da MEF é conseguida com a mesma aproximação como quando desenvolvida em *software*. As alterações em tempo real também são admitidas. Isto permite o controlo de algoritmos a ser executados já que de outra forma não seria possível devido aos recursos do PLD não serem suficientes. É conseguido somente mantendo a parte da MEF que afecta o comportamento actual dentro do PLD, isto é, a parte que é requerida para a execução a qualquer momento, enquanto a configuração das outras partes é mantida na memória externa. Como a execução evolui, outras partes são carregadas no PLD, e as partes que já não são necessárias são descarregadas (de forma a libertar recursos). Neste capítulo será possível mostrar que estas técnicas podem ser implementadas numa reconfiguração em FPGAs. A ideia principal da aproximação considerada é a combinação das potencialidades da FPGA com alguns modelos propostos de MEFs junto com métodos para produzir especificação que se vão alterando.

5.2 Modelo de Funcionamento

O HT é um circuito possível de modificar, mas que já foi implementado numa FPGA. Por outras palavras, a FPGA já foi configurada mas continua com a possibilidade de ser reconfigurada através da reprogramação de alguns blocos do HT. Isto permite reduzir significativamente o tempo de reconfiguração e manter todos os tempos predefinidos no circuito.

Assim, por um lado o HT é um circuito com uma aplicação específica, mas por outro, permite implementar diferentes funcionalidades que são fornecidas para que as características básicas do circuito sejam optimizadas.

Para construir uma MEF de uma especificação dada, podem-se usar vários métodos tal como sugerido em [23]. Estes são baseados no modelo de MEF em *hardware* e permitem que os circuitos sejam sintetizados. Os mecanismos de interacção entre módulos numa especificação hierárquica podem ser, implicitamente, aplicados através de níveis hierárquicos diferentes, que são associados a registos de uma pilha de memória. Por outro lado, pode-se especificar explicitamente a interacção entre pequenas MEF associadas a módulos. MEF com comportamento modificável pode ser construído baseado em reconfiguração dinâmica [24] e estática [25] em FPGA. Para o último caso pode-se utilizar um modelo de MEF baseado em blocos RAM/ROM [26]. Do ponto de

vista do estado de codificação, diferentes modelos de uma MEF podem ser classificados da seguinte forma:

- MEF baseadas em codificação de estado binário, na qual, normalmente, é utilizado R = int $\log_2$ M, sendo R o tamanho do registo de estado, e M são o número dos estados;
- MEF baseadas em codificação de estado *one-hot*, onde R=M;
- MEF com estados implícitos, no qual o registo de saída mantém os valores de saída y_1, …, y_N, também é considerado o registo de estado, isto é, os estados e os registos de saída são fundidos [27];
- MEF baseadas na codificação de estado *fuzzy* [25, 26], onde R≥ int $\log_2$M e códigos para estados diferentes podem ter tamanhos diferentes, isto é, os estados a_m e ($a_m \neq a_s$) tem códigos de tamanho R_m e R_s, e geralmente $R_m \neq R_s$.

Fazendo a análise de vários modelos com estratégias para codificação de estado com resultados experimentais apresentados em [27] optou-se por uma MEF com ligação em cascata.

A Figura 5.1 descreve uma implementação em hardware de uma MEF reprogramável. É composta por dois blocos baseados em memória RAM e de um registo (Rg). O *multiplexer* programável (PM), no qual a estrutura é mostrada na Figura 5.1b, permite que qualquer entrada $x_j \in \{x_1,\ldots, x_L\}$ seja seleccionada de tal forma que $p=x_j$ e o valor j pode ser especificado pelo *multiplexer* (MRAM). Por exemplo, se os códigos da MEF forem valores binários e se a entrada x_3 afectar as transições do estado a_1 (R=3), então o endereço 001 da MRAM atribui saídas 010 para o *multiplexer* 8:1. Claramente pode-se fazer corresponder os estados a_0, …, a_{M-1} às entradas x_1, …, x_L. A transição de estados na RAM (STRAM - Static Transition Random Access Memory) permite-nos gerar códigos para os estados e saídas seguintes. Por exemplo, se R=3 e se houver uma transição do estado de a_2 apresentado na Tabela 5.1 então no endereço 0100 STRAM contêm o código (D_1, …, D_3) = 000 e no endereço 0101 o código (D_1, …, D_3) = 100. Obviamente todo o subconjunto dos sinais de saída y_1, …, y_N pode ser gerado arbitrariamente para qualquer transição do estado.

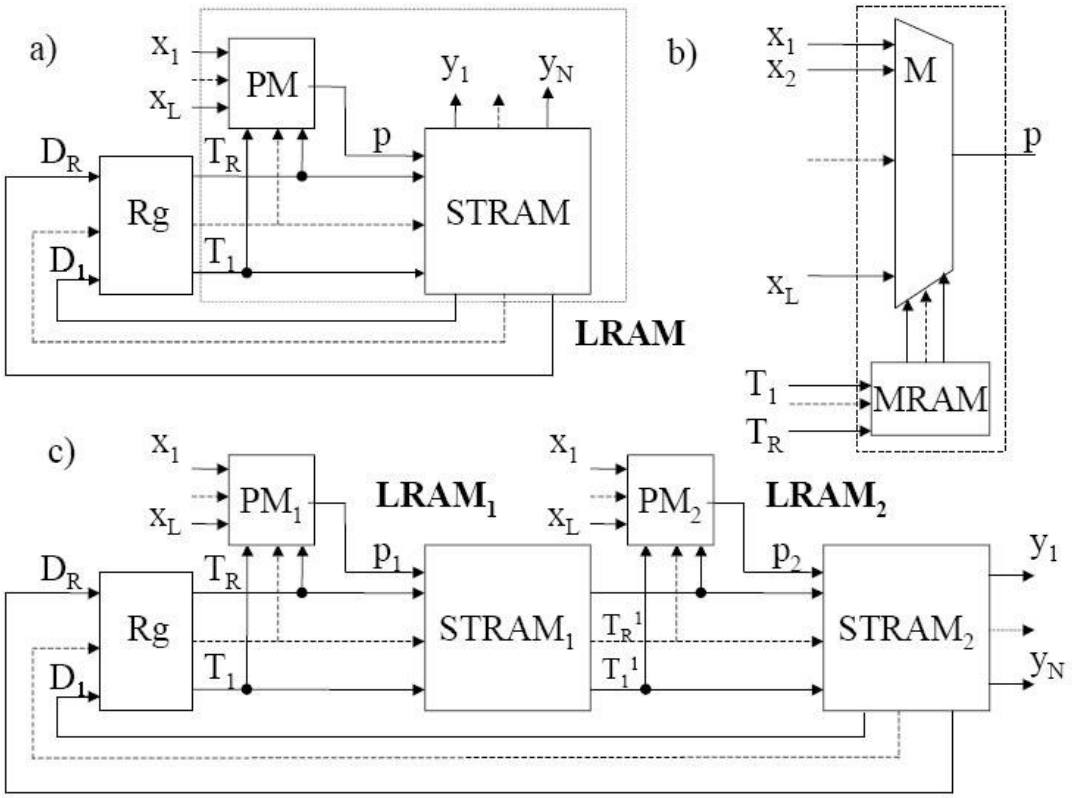

Figura 5.1 – Modelo da MEF reprogramável (a), multiplexer programável (b), modelo em cascata da MEF reprogramável (c) [22]

Modificando os índices de MRAM e de STRAM pode-se implementar qualquer comportamento na MEF que se pretenda (dentro dos valores predefinidos, tais como R, L, e N). Entretanto os circuitos da Figura 5.1 (a) e (b), tem uma limitação muito importante: qualquer transição apenas pode ser afectada por uma única variável de entrada. Uma transição arbitrária do estado pode ser alterada de tal forma que permita esta limitação ser satisfeita dividindo as transições do estado e inserindo estados *dummy*. Um exemplo, é mostrado na Figura 5.2 para as transições do estado a_0 (ver Tabela 5.1). Entretanto, no caso geral, isto altera o comportamento da MEF, aumenta o número dos estados M, e reduz a velocidade de transições dos estados. Para muitas aplicações práticas esta técnica não pode ser empregue.

Tabela 5.1 – Tabela de transições de estados

a_{from}	$K(a_{from})$	$X(a_{from},a_{to})$	a_{to}	$K(a_{to})$
a_0	000	(not x_1)(not x_2)	a_0	000
		(not x_1)x_2	a_1	001
		x_1x_2	a_2	010
		x_1(not x_2)	a_3	011
a_1	001	x_3	a_1	001
		(not x_3)x_4	a_3	011
		(not x_3)(not x_4)	a_4	100
a_2	010	(not x_5)	a_0	000
		x_5	a_4	100
a_3	011	x_6	a_2	010
		(not x_6)	a_3	011
a_4	100	(not x_1)	a_3	011
		x_1	a_4	100

O modelo proposto de MEF reprogramável em cascata (ver Figura 5.1c) permite resolver este problema. Seguramente o circuito descrito na Figura 5.1a, permite implementar qualquer transição de estado, causado por uma entrada, e o próximo estado pode ser um qualquer estado, que é requerido por comportamento ou por um estado *dummy* (ver estados na Tabela 5.2). O último é usado apenas para proporcionar a reconfigurabilidade e não deve ser armazenado no registo da MEF. Assim as saídas dos blocos reprogramáveis do primeiro nível (nível RAM ou LRAM) têm que ser ligados com as entradas de LRAM similares para o nível seguinte. As saídas do LRAM do último nível mudam o estado do registo da MEF.

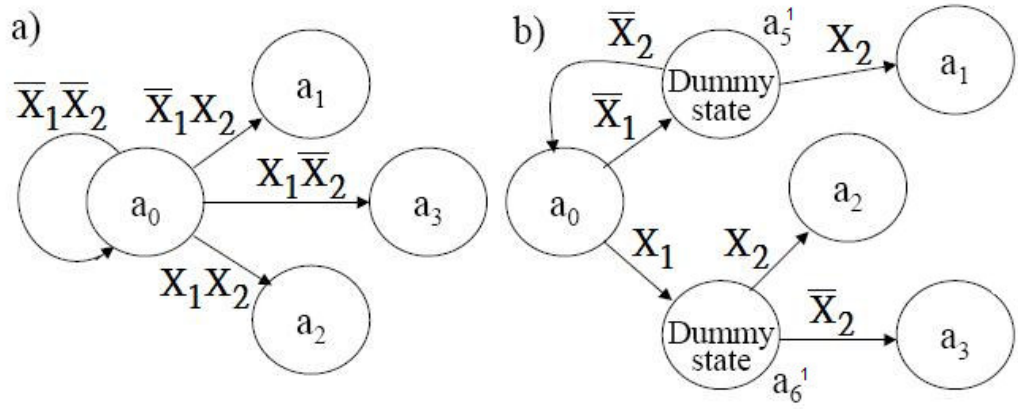

Figura 5.2 – Parte do grafo de transição de estados na MEF (a) e divisão das transições (b).

Assim toda a transição de direcção do estado pode ser realizada sem qualquer limitação específica. Os estados *dummy* podem aparecer somente entre LRAMs durante a propagação dos respectivos sinais no circuito combinatório composto por LRAMs. Assim praticamente não se reduz a velocidade de transição dos estados. As saídas das MEF podem ter qualquer forma, quer para a MEF de Moore ou de Mealy. Mesmo um modelo Moore/Mealy misto [28] pode ser usado. A capacidade de alterar o HT considerado acima para a MEF, é fornecida através da reprogramação dos blocos RAM/ROM.

Tabela 5.2 – Tabela de transições com divisão de estados

a_{from}	$K(a_{from})$	$X(a_{from},a_{to})$	a_{to}	$K(a_{to})$
a_0	000	(not x_1) x_1	a_5^1 a_6^1	101 110
a_5^1	101	(not x_2) x_2	a_0 a_1	000 001
a_6^1	110	x_2 (not x_2)	a_2 a_3	010 011
a_1	001	x_3 (not x_3)	a_1 a_7^1	001 111
a_7^1	111	x_4 (not x_4)	a_3 a_4	011 100
a_2	010	(not x_5) x_5	a_0 a_4	000 100
a_3	011	x_6 (not x_6)	a_2 a_3	010 011
a_4	100	(not x_1) x_1	a_3 a_4	011 100

5.3 Implementação

Existem muitos sistemas digitais que exigem operações sobre vectores booleanos, que podem ter um tamanho arbitrário e o número de operações possíveis sobre tais vectores é infinito (pelo menos muito grande). Por outro lado, em regra, cada aplicação específica requer um número muito limitado de operações. Assim, é razoável que se construa um circuito reutilizável e que se execute um número limitado de operações sobre vectores booleanos, mas estas operações podem ser personalizadas para uma utilização de um número ilimitado de aplicações.

Por exemplo, existem muitas aplicações práticas que precisam resolver vários

problemas combinatórios [29] e [30]. Normalmente este tipo de problemas é muito moroso. Por isso é que foram concebidos, muitos e diferentes, aceleradores em hardware [31], [32], [33], etc.

O HT foi implementado com o auxílio do software ISE da Xilinx Foundation. Como se pode verificar através da Figura 5.1, o HT foi construído por *multiplexers* programáveis (PM – Programmable Multiplexer) e por memórias RAM. Estas têm duas funções, que passam por configurar o *multiplexer* (MRAM - Multiplexer Random Access Memory) e possibilitar as transições dos estados (STRAM – State Transition RAM). Todos os blocos foram criados em *VHDL*. As alterações do PM são conseguidas através da modificação do ficheiro de configuração que é necessário para as MRAMs e STRAMs. Para uma dada MEF, o ficheiro de configuração será diferente dependendo da aplicação prática pretendida. Para alterar o comportamento é necessário programar os blocos de memória.

```
0000000010000
0000000010001
0000000010000
0000000010010
0000000010000
0000000010011
0000000010000
0000000010011
0000000010000
0000000010000
0000000010000
0000000010000
0000000010000
0000000010000
0000000010000
0000000010000
0000000100000
0000000100000
0000000100001
0000000100001
0000000100010
0000000100010
0000000100011
0000000100011
• • •
```

WE OUTPUT RAM	WE MRAM4	WE MRAM3	WE MRAM2	WE MRAM1	WE STRAM3	WE STRAM2	WE STRAM1	DATA
1 bit	1 bit	1 bit	1 bit	1 bit	1 bit	1 bit	1 bit	4 bits

Figura 5.3 – Ficheiro de configuração das memórias

Na Figura 5.3 pode-se ver parte do ficheiro de configuração do HT. Na parte inferior da figura é possível observar o significado de cada coluna. Basicamente, existem 4 bits de dados tendo os restantes a função de activar a RAM a ser escrita. Activando uma dada RAM para escrita, ela passa a armazenar os dados, que se encontram nas colunas mais à direita do ficheiro.

Foi criada pelo Bruno Pimentel (aluno de doutoramento) uma aplicação, denominada de HT based EU Examples (EU - Execution Unit), na linguagem C#, que permite enviar o ficheiro de configuração das memórias e enviar/receber as entradas e saídas da MEF. Esta aplicação contém apenas interfaces para duas MEFs, já que o objectivo principal não é a MEF em si, mas sim o método de configurar o HT.

Inicialmente, é enviado o ficheiro de configuração que, a FPGA ao receber, vai configurar o HT. Depois de a MEF estar criada, ela envia o estado de saída para a aplicação a correr no computador e a MEF fica pronta a receber as entradas, que serão enviadas pela aplicação.

A comunicação existente entre o computador e a placa DETIUA-S3 é realizada através de *bluetooth*. No computador existe um *dongle bluetooth*, que permite comunicar com o módulo bluetooth existente na placa. A aplicação vê o *dongle Bluetooth* como uma porta série, sendo necessário escolher a porta série à qual está associado.

Através da Figura 5.4 é possível verificar os passos necessários para implementar uma MEF, baseada em HT, e sua utilização prática.

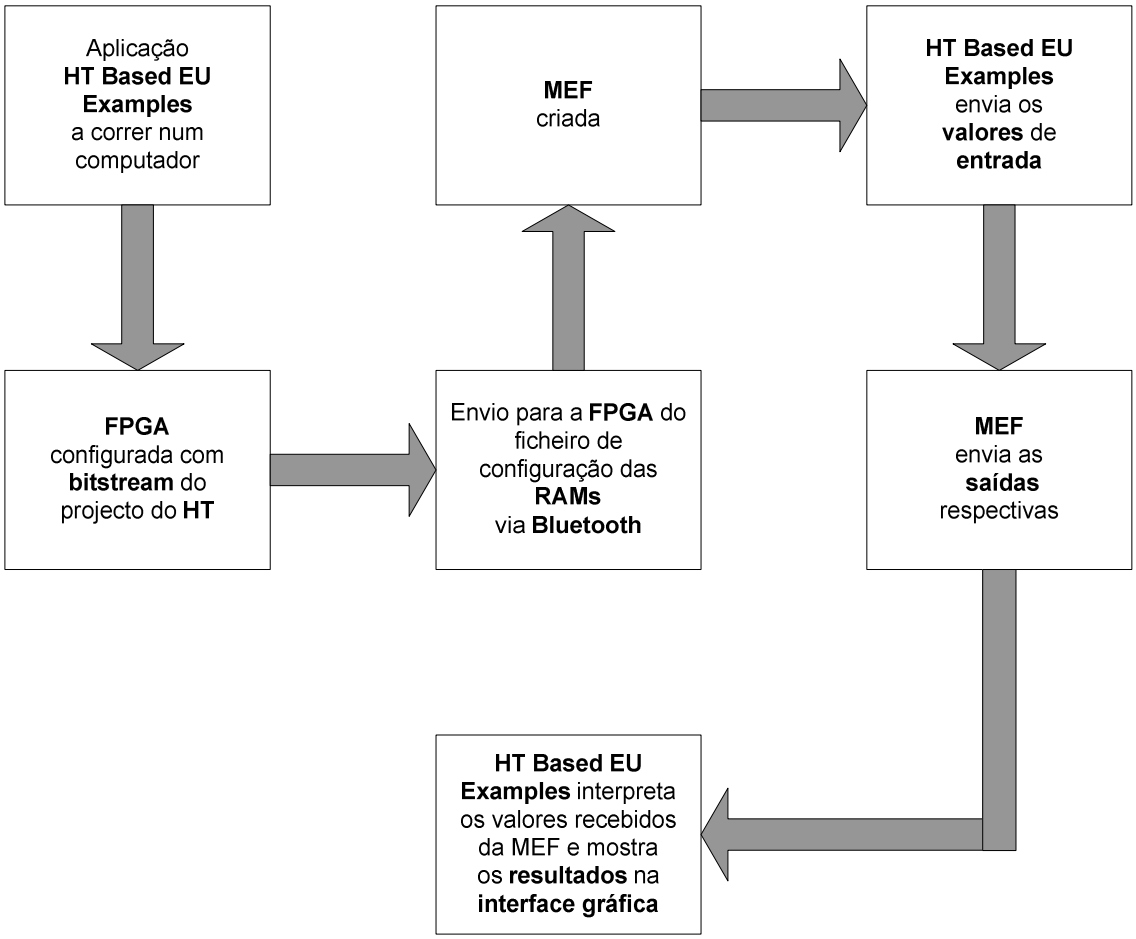

Figura 5.4 – Sequência de passos para utilização de uma MEF

A aplicação HT based EU Examples (Figura 5.5 e Figura 5.6) foi desenvolvida de forma a implementar um grande número de interfaces que permite resolver um grande número de problemas. Ela foi desenvolvida para permitir a introdução dos dados a processar, observar alguns dos estados da MEF e obter o resultado desse processamento. Também permite escolher, através dos separadores, a interface que se pretende utilizar. Existe o botão *Program* que tem como função enviar o ficheiro de configuração do HT.

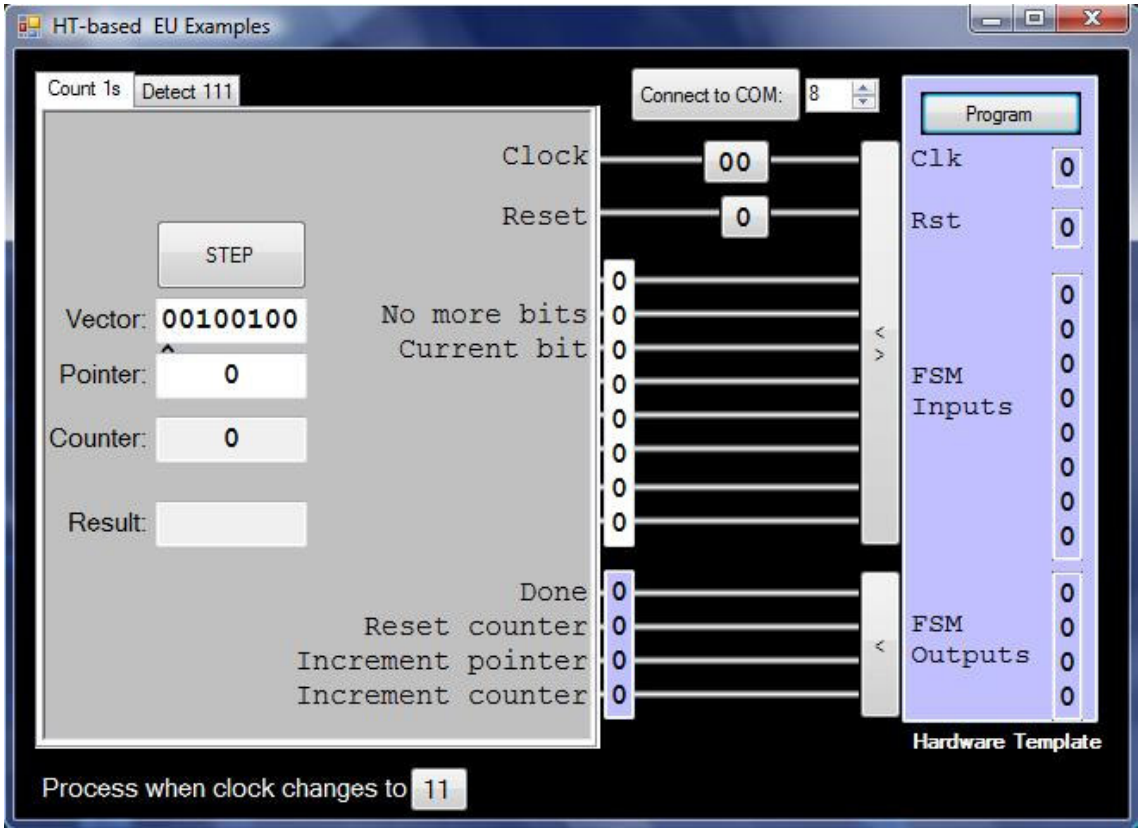

Figura 5.5 – HT-based EU Examples – Count 1s

Na Figura 5.5 encontra-se uma imagem da aplicação HT-based EU Examples contendo a interface Count 1s em evidência. Esta interface permite saber quantos "uns" se encontram num dado vector. Tal como foi referido anteriormente, a interface computador placa DETIUA-S3 é realizada através de *bluetooth*. A aplicação HT-based EU Examples tem um campo que permite introduzir qual a porta associada ao *dongle bluettoth* e posteriormente fazer a ligação. Esta aplicação contém alguns campos que na realidade não dá uma informação essencial, mas que é interessante quer numa primeira fase na detecção de erros quer para detectar passos intermédios.

É possível fazer o *reset* ao sistema, isto é, colocar o sistema no estado inicial limpando todos os campos com excepção do vector que se quer verificar. No campo *Clock* é possível introduzir o número do ciclo de relógio, para correr a aplicação passo a passo. O campo *Pointer* permite ver qual o elemento do vector que está a ser verificado pelo sistema, o *Counter* dá a informação do número de uns que verificou num dado momento, enquanto que o *Result* permite ver o número total de uns que continha o vector.

É possível ver também os valores de entrada e saída da MEF.

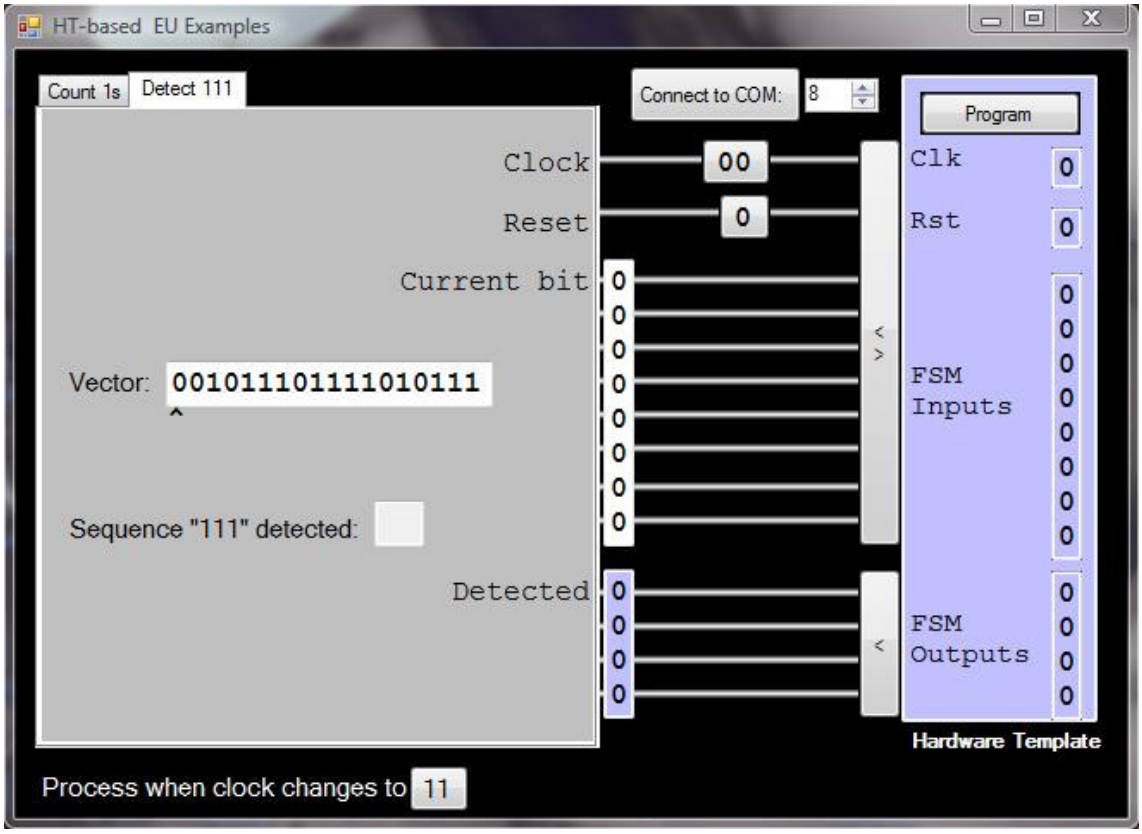

Figura 5.6 – HT-based EU Examples – Detect 111

Através da Figura 5.6 é possível observar a interface que permite detectar o número de sequências de uns consecutivos num dado vector.

A interface funciona da mesma forma da descrita anteriormente, mas neste caso tem somente dois campos. Um permite introduzir o vector a ser processado e o outro o número de uns consecutivos que se pretende achar no vector de entrada.

É possível implementar uma MEF com vários comportamentos. Esses comportamentos são obtidos através da configuração da MEF. É de salientar, que esse comportamento pode ser alterado sem a necessidade de alterar a MEF. Na Figura 5.7 encontra-se um exemplo de uma MEF configurada com um dado comportamento. Para ser alterado, esse comportamento, é necessário enviar o ficheiro com os valores que programam as respectivas memórias RAM.

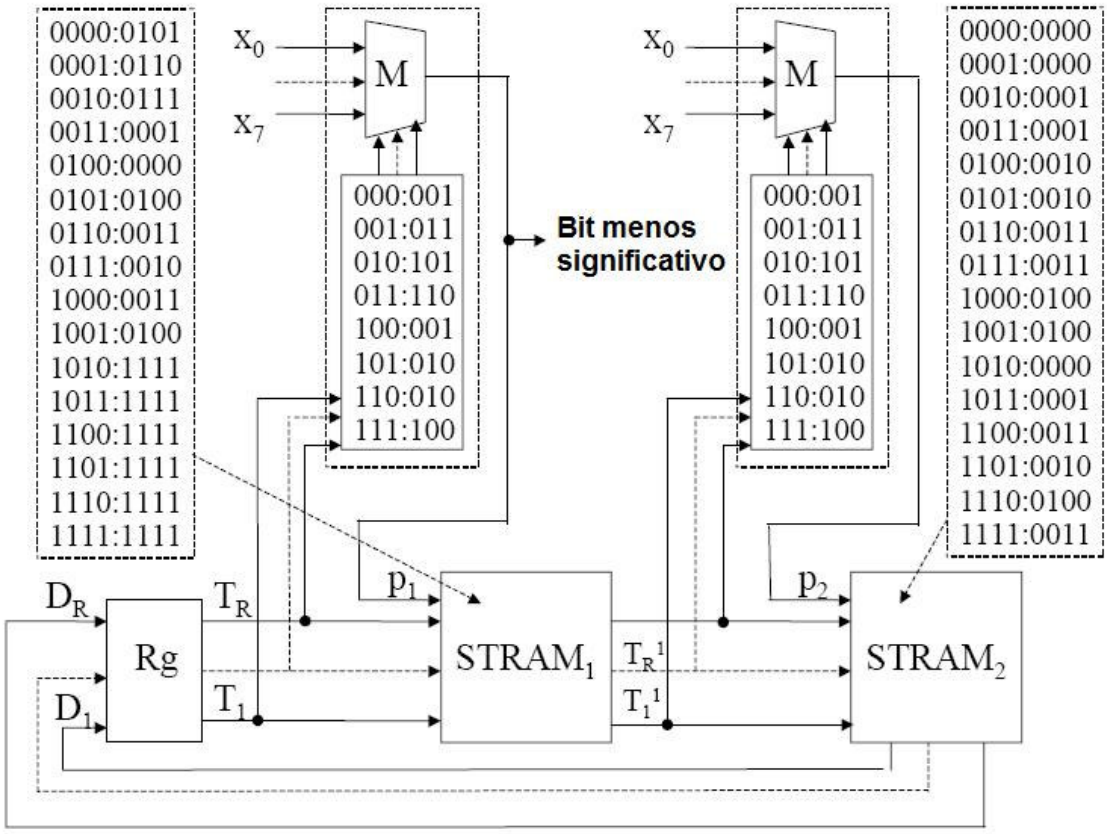

Figura 5.7 – Implementação de uma MEF reconfigurável com um dado comportamento [22]

5.4 Conclusões

Neste capítulo foi possível mostrar uma MEF que possibilita a alteração do seu comportamento e a sua utilização para uma série de aplicações práticas, apesar de implementação de tarefas simples. É mostrado que qualquer MEF reconfigurável pode ser construída para que ela possa ser utilizada por um conjunto de aplicações para determinadas áreas específicas tais como, cálculo combinatório, controladores de rede, etc.

Foi possível avaliar uma das potencialidades deste método que passa por alterar o comportamento da MEF sem a necessidade de se alterar a lógica inicial.

Capítulo 6

Extensões da Placa de Prototipagem para Resolver Vários Problemas em Engenharia

Sumário

Neste capítulo vão ser descritos vários módulos/placas que foram desenvolvidos por mim e por alunos que utilizaram a placa DETIUA-S3, durante este processo disponibilizei todo o apoio necessário para a execução dos mesmos. Esse apoio traduziu-se quer em esclarecimento relativo à placa DETIUA-S3 e sua funcionalidade, quer ao desenvolvimento do projecto na ferramenta ORCAD da Cadence®, que foram desenvolvidos esses projectos.

6.1 Introdução

A placa de prototipagem, tal como foi referido no capítulo 2, não possui periféricos associados. Existem diversas situações que exigem um ou mais periféricos. No processo educativo, tal como já foi referido anteriormente, existe essa necessidade, podendo existir ao mesmo tempo, diferentes grupos de pessoas, que podem utilizar a placa DETIUA-S3 juntamente com um ou vários periféricos. A existência dessa necessidade levou ao desenvolvimento de módulos/placas que contêm esses mesmos periféricos [1, 18]. Esses módulos/placas são ligados aos barramentos existentes na placa DETIUA-S3.

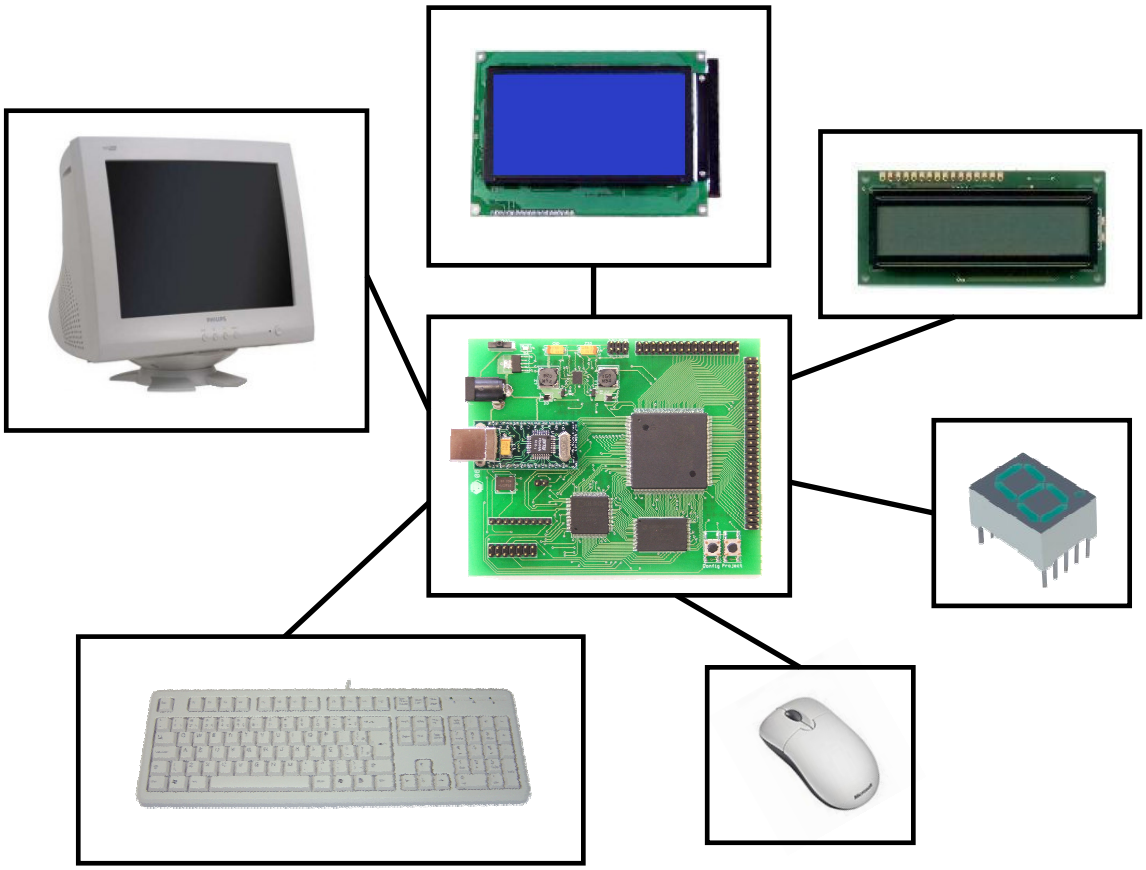

Figura 6.1 – Periféricos para interligação à placa DETIUA-S3

Na Figura 6.1 é possível observar alguns dos periféricos possíveis de integrar na placa DETIUA-S3. Devido à existência de dois barramentos com vários pinos de entrada/saída, é possível integrar mais do que um periférico simultaneamente.

A placa DETIUA foi utilizada pelos alunos de LECT/MIECT e LEET/MIEET das disciplinas Sistemas Digitais Reconfiguráveis e Computação Reconfigurável.

6.2 Dispositivos e placas de extensão

A integração de periféricos externos são extremamente importantes e por isso aqui serão relatados e explicados alguns dos periféricos mais comuns. Além disso são mostradas algumas placas de extensão que contém esses mesmos periféricos.

6.2.1 Interacção com monitor VGA, teclado e rato PS/2.

Neste ponto serão descritos os princípios de funcionamento de um monitor VGA, de um teclado e rato PS/2. Além disso será descrita a placa de extensão que contempla os três periféricos referidos.

6.2.1.1 Monitor VGA

Um monitor VGA recebe a informação de cor com a ajuda de 3 sinais: R (vermelho), G (verde) e B (azul). Cada um destes sinais controla a emissão de electrões, que por sua vez iluminam um ponto na superfície do ecrã com uma cor primária. Os níveis analógicos destes sinais especificam a intensidade de cada uma das cores primárias podendo variar entre 0 V (completamente escuro) e 0.7 V (brilho máximo).
Uma das possibilidades é controlar as saídas de cor analógicas por uma DAC que possui 6 entradas digitais (ver Figura 6.2).
Sendo assim, torna-se possível especificar 4 níveis de cada cor (de 00 a 11) e representar na totalidade 64 cores diferentes (2^6).

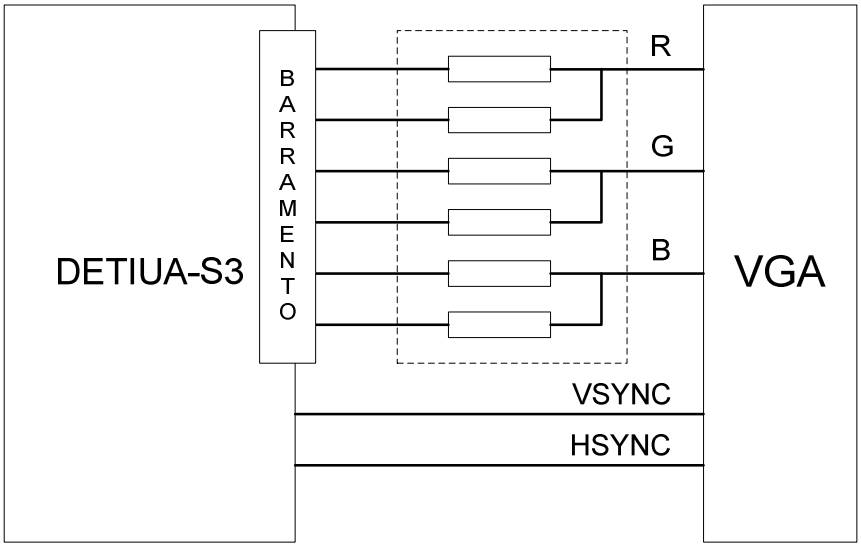

Figura 6.2 – Ligação dos pinos da FPGA que fornecem a informação de cor e de sincronização com a saída VGA da placa DETIUA-S3.

Uma imagem no ecrã do monitor é composta por “m” linhas cada uma da qual contém ”n” pixels. Para implementar o controlador usamos a resolução de n×m = 640×480 pixels. Para desenhar uma imagem o feixe de electrões percorre o ecrã de esquerda para a direita e de cima para baixo.

Em [34] encontra-se uma descrição pormenorizada das características temporais dos sinais VGA, a especificação do algoritmo para visualização de uma imagem no ecrã e a implementação do controlador utilizado na placa DETIUA-S3.

Os blocos representados na Figura 6.3 foram especificados em VHDL, e constituem o controlador VGA que funciona no modo de texto.

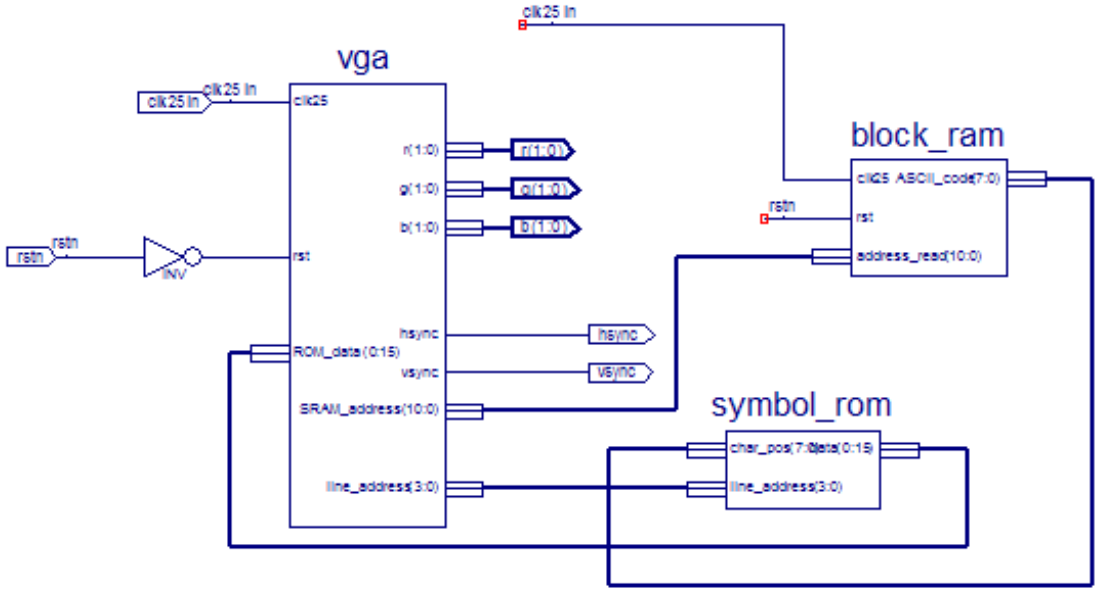

Figura 6.3 – Esquema de blocos que controla um monitor VGA

6.2.1.2 Teclado e rato PS/2

O teclado e rato PS/2 utilizam um protocolo série síncrono e bidireccional. O barramento está "*Idle*", quando ambas as linhas estão no nível lógico '1'. Este é o único estado em que o teclado/rato inicia a transmissão de dados. O anfitrião tem o controlo final sobre o barramento e pode impedir a comunicação, a qualquer momento, ao colocar o sinal de relógio no nível lógico '0'.

É o dispositivo que gera o sinal de relógio. Se o anfitrião quer enviar dados, ele deve primeiro inibir a comunicação do dispositivo, colocando o sinal de relógio no nível lógico '0'. O anfitrião, de seguida, coloca o sinal de dados no nível lógico '0' e produz o sinal de relógio. Este é o estado " Request-to-Send" e sinaliza o dispositivo para iniciar o sinal de relógio. Na tabela Tabela 6.1 é possível verificar os estados possíveis que o barramento pode ter.

Tabela 6.1 – Sumário do estado do barramento

Estado	Nível lógico	
	Dados	**Relógio**
Idle	1	1
Communication Inhibited	1	0
Host Request-to-Send	0	1

Os dados são transmitidos bit a bit, e é enviado numa trama de 11 ou 12 bits. A trama é constituída por 1 start bit, 8 bits de dados, sendo enviado primeiro o menos significativo, 1 bit de paridade (paridade ímpar), 1 stop bit e 1 bit de *acknowledge* (apenas comunicação do anfitrião para o dispositivo).

O bit de paridade é ajustado, se houver um número par de "uns" nos bits de dados e é colocado a zero, se houver um número ímpar de "uns" nos bits de dados. É sempre adicionado "uns" ao conjunto de bits de dados, mais o bit de paridade, até um número ímpar (paridade ímpar). Utiliza-se para detectar erros. O teclado/rato deve verificar esse bit e se for incorrecto ele deve responder como se tivesse recebido um comando inválido.

Os dados enviados pelo dispositivo ao anfitrião são lidos na transição descendente do sinal de relógio, enquanto os dados enviados pelo anfitrião para o dispositivo são lidos na transição ascendente. A frequência de relógio deve estar compreendida entre os 10 e os 16,7 KHz. Isto significa que o relógio deve estar no nível lógico '1' entre 30 a 50 µs e no nível lógico '0' entre 30 a 50 µs [35].

6.2.1.3 Placa de extensão que inclui saída VGA, teclado e rato PS/2

Foi desenvolvida uma placa (ver Figura 6.4) que contém uma saída VGA, que possibilita a ligação de um monitor, e duas fichas PS/2, permitindo ligar um teclado e um rato.

A placa foi desenvolvida para o processo educativo permitindo aos alunos, numa primeira fase, criar o controlador para cada um dos dispositivos, e posteriormente interligá-los.

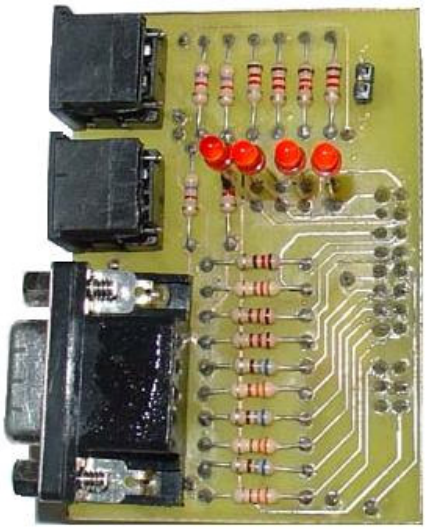

Figura 6.4 – Placa de extensão que suporta monitor VGA, teclado e rato PS/2

6.2.2 Interacção com botões de pressão, interruptores e LEDs

Alguns dos periféricos mais básicos de utilizar são os botões de pressão, interruptores e LEDs. Eles permitem efectuar controlos que, em muitas situações, são extremamente úteis e simples.

Foram desenvolvidas algumas placas contendo botões de pressão, interruptores e LEDs. Por exemplo, uma das placas desenvolvidas permite visualizar através de LEDs o resultado de operações aritméticas, sendo os valores de entrada introduzidos através de interruptores.

Existem outras placas que contêm estes periféricos, tal como a placa DETIUA-S3 que inclui dois botões de pressão e um LED.

6.2.3 Interacção com LCD e mostrador de 8 segmentos

6.2.3.1 LCD

Um LCD é um periférico de saída que permite visualizar um conjunto de caracteres, que neste caso se encontram previamente guardados numa ROM existente no módulo.

É possível verificar através da Tabela 6.2 a descrição e o número dos pinos do LCD utilizado. Este LCD contém 2 linhas de 16 caracteres cada.

Tabela 6.2 – Descrição dos pinos do LCD

Pino	Simbolo	Função	Pino	Simbolo	Função
1	VSS	Alimentação 0V	8	D1	Dados
2	VDD	Alimentação +5V	9	D2	Dados
3	VEE	Contraste	10	D3	Dados
4	CS	1'-Comando, '0'- Dados	11	D4	Dados
5	R/W	1'-Leitura, '0'- Escrita	12	D5	Dados
6	E	Enable	13	D6	Dados
7	D0	Dados (LSB)	14	D7	Dados (MSB)

Na Figura 6.5 encontra-se o esquema de blocos do controlador do LCD da marca Hitachi. O bloco que tem o nome hitachi_lcd foi desenvolvido em VHDL.

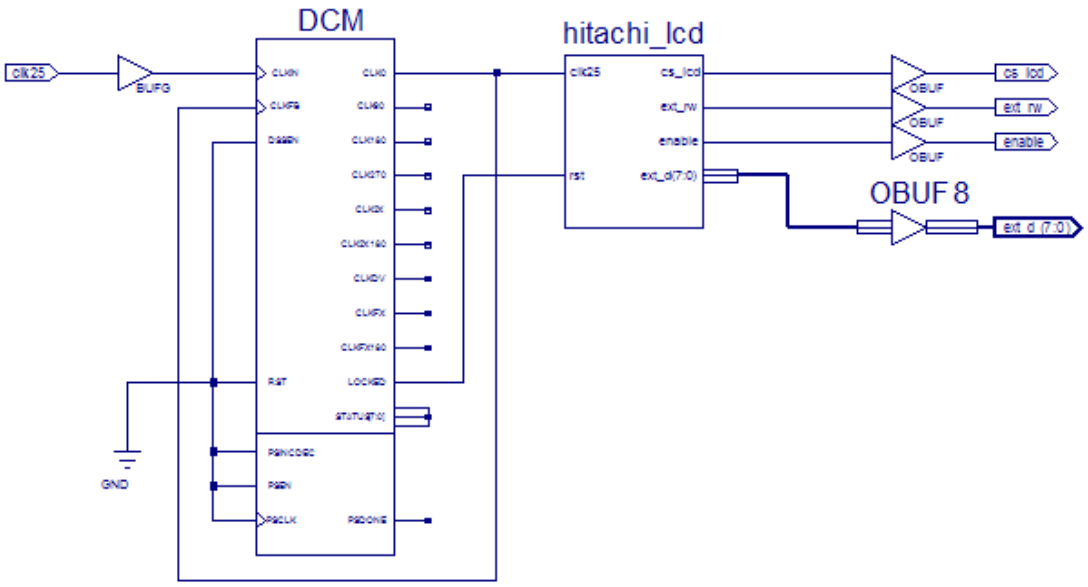

Figura 6.5 – Esquema de blocos para controlo do LCD

6.2.3.2 Mostrador de 8 segmentos

Um mostrador de 8 segmentos, tal como o nome indica, é um agrupamento de sete LEDs (ver Figura 6.6). Existe ainda um oitavo segmento (DP-Decimal Point) que é o ponto e que permite a visualização de números não inteiros.

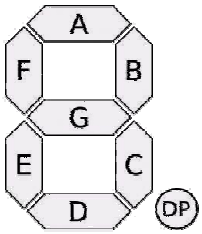

Figura 6.6 – Mostrador de 8 segmentos

6.2.3.3 Placas de extensão que incluem LCD e mostrador de 8 segmentos

Foram desenvolvidas algumas placas que incluem um ou vários mostradores de 8 segmentos e LCDs. Alguns exemplos estão representados na Figura 6.7, Figura 6.9 e Figura 6.10.

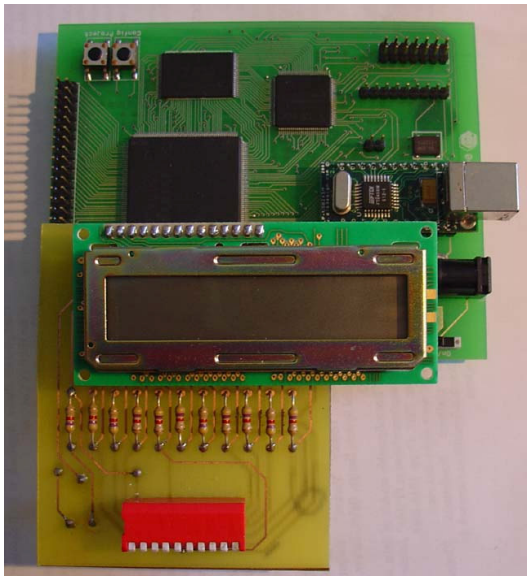

Figura 6.7 – Placa de extensão utilizando um LCD

A placa da Figura 6.7 foi desenvolvida com o objectivo de efectuar um conjunto de operações aritméticas. Nos interruptores são introduzidos combinações que contêm os dois operandos e também a operação a efectuar. No LCD aparece a expressão ordenada da seguinte forma: Operando1 Operação Operando2 = Resultado. Na Figura 6.8 encontra-se os blocos, implementados em VHDL, necessários para a realização desta

tarefa. Facilmente se pode verificar que esta placa não foi desenvolvida exclusivamente para a tarefa referida, mas sim, para ser utilizada em várias tarefas diferentes.

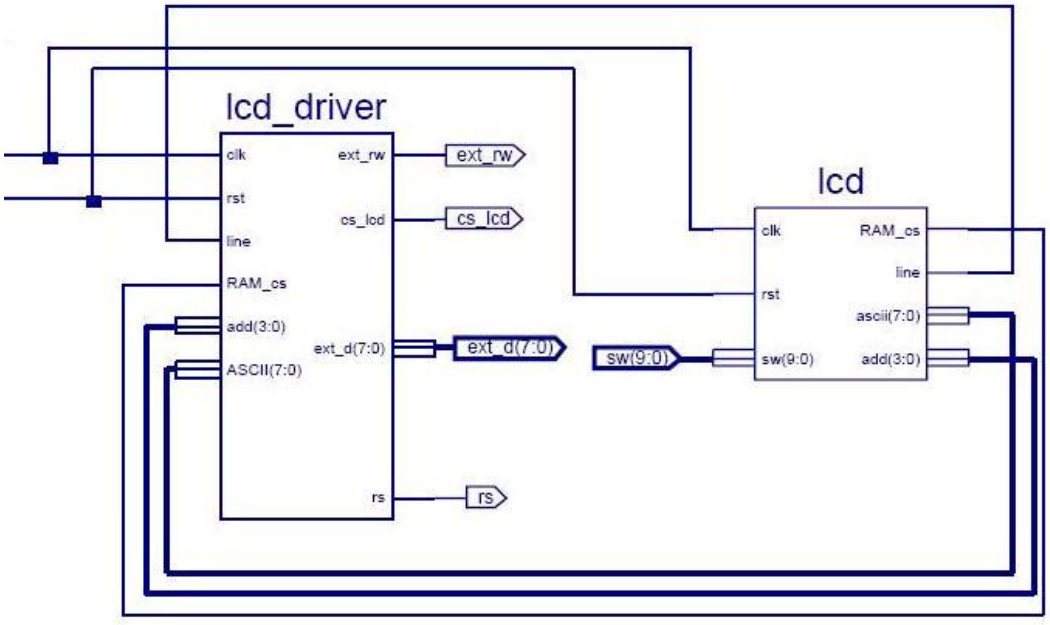

Figura 6.8 – Esquema dos blocos de controlo do LCD

Na Figura 6.9 encontra-se uma placa de extensão, que permite a realização de várias tarefas tais como: através de uma combinação nos interruptores e através de um descodificador implementado na FPGA, visualizar os valores desejados nos mostradores de 8 segmentos; implementação de um contador; implementação de um relógio digital, no qual aparece as horas e os minutos, etc.

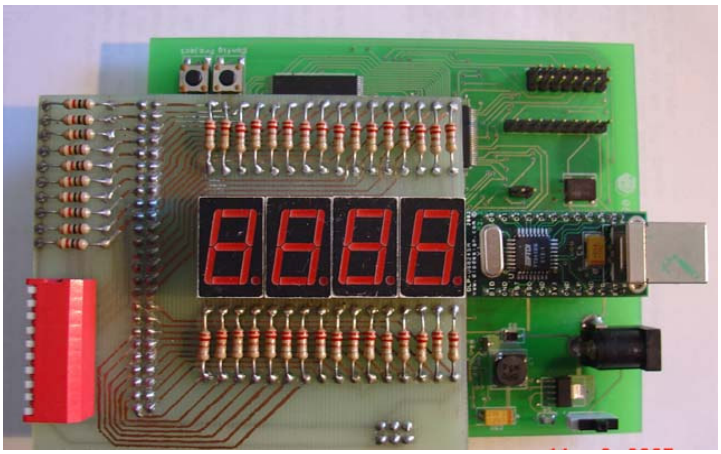

Figura 6.9 – Placa de extensão usando 4 mostradores de 8 segmentos

A placa de expansão que está na Figura 6.10, exibe os mostradores de 8 segmentos a

indicar o valor da diferença de potencial medida de um dado dispositivo.

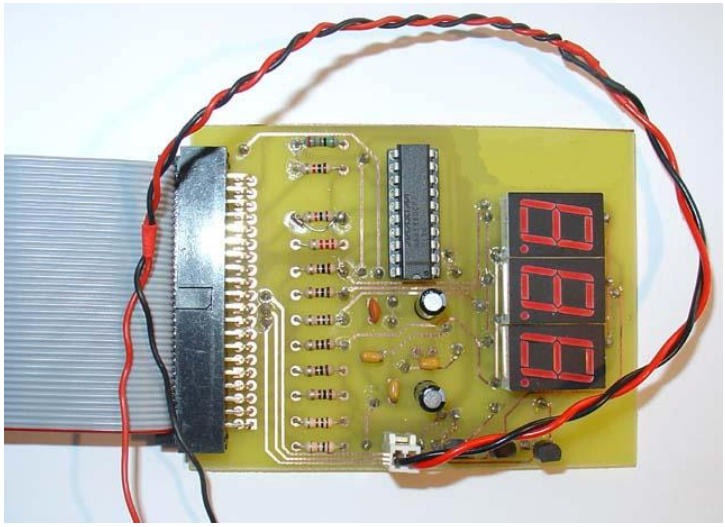

Figura 6.10 – Placa de extensão usando 3 mostradores de 8 segmentos – voltímetro

6.2.4 Comunicação com porta série (RS-232)

Desenvolveu-se um módulo que implementa o protocolo de comunicações no qual os bits de dados são enviados em série assincronamente (Figura 6.11). A cada bit corresponde um tempo predefinido através do *baud-rate* acordado entre dois (ou mais) dispositivos. Possui limitações quer de velocidade, quer a nível de distâncias. É um protocolo que tem tendência a desaparecer. Foi implementada uma versão reduzida do protocolo (ignorando o *hand-shacking* e a paridade, bem como os sinais de controlo).

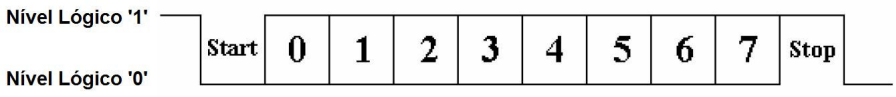

Figura 6.11 – Protocolo série

Utilizou-se apenas as linhas de *Signal Ground* (massa - pino 5), *Received Data* (RD - pino 2) e *Transmitted Data* (TD - pino 3), não se utilizou paridade, usou-se um start bit e um stop bit. Para adaptação entre os níveis de tensão usados no PC (-12V="1"e "+12V="0") e na FPGA (+3.3V="1"e 0V="0") utilizou-se um MAX3232 (Figura 6.12) [36].

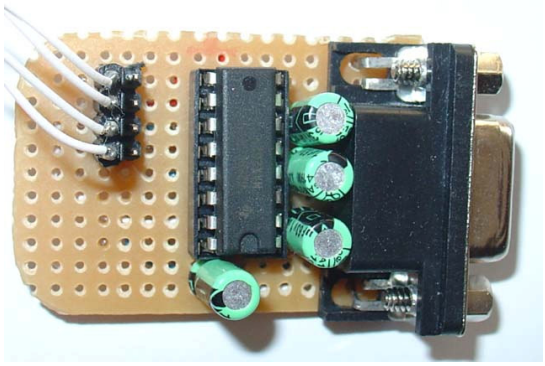

Figura 6.12 – Módulo RS232

Foi desenvolvido um projecto de teste que possibilita a troca de informação, entre o computador e a placa DETIUA-S3. O projecto desenvolvido no ISE da Xilinx está representado na Figura 6.13.

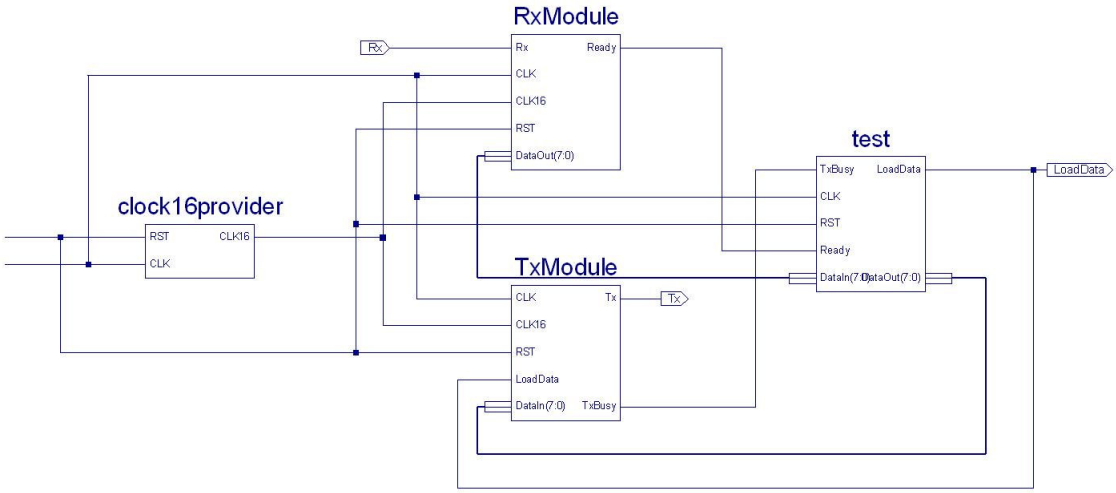

Figura 6.13 – Esquemático do projecto RS232

Os vários módulos integrantes do projecto foram desenvolvidos em VHDL. O bloco *clock16provider* fornece o relógio para um *baud-rate* de 9600, enquanto que os restantes blocos, tal como o nome indica, têm implementado o módulo de transmissão *TXModule*, módulo de recepção *RXModule* e um bloco de teste.

6.3 Sistemas utilizando periféricos externos

A Figura 6.14 apresenta um exemplo, que é muito útil para fins educativos. Ele demonstra como implementar um microcomputador simples, com base nos métodos e ferramentas propostas. O microcomputador é implementado a partir da placa DETIUA-

S3, do conector de expansão e alguns periféricos, aos quais podem ser associados, tais como, um monitor VGA, um teclado, um rato, e porta série. Desenvolvidas as interfaces que são necessárias para a comunicação com os periféricos, posteriormente, apenas é necessário implementar os circuitos para o microcomputador. As interfaces estão numa biblioteca que depois são utilizadas como um bloco que controla uma dada interface e que em certas situações não é necessário conhecer a sua implementação.

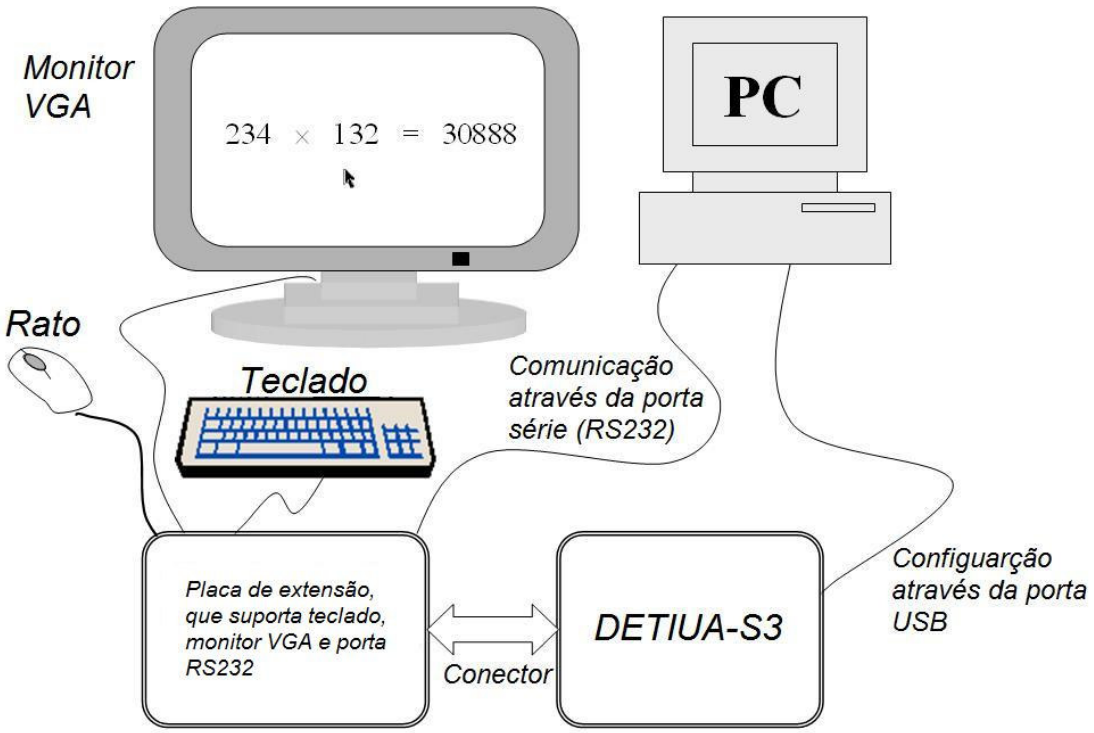

Figura 6.14 – Exemplo de um sistema usando VGA, teclado, rato e porta RS232

A Figura 6.14 apresenta um exemplo muito simples de um microcomputador que permite realizar operações aritméticas. Existe um grande número de aplicações possíveis de implementar.

Através dos dispositivos externos à placa, descritos anteriormente, pode-se implementar um grande número de sistemas. Existe no mercado, em abundância, dispositivos e módulos possíveis de integrar na placa DETIUA-S3. Módulos para rádio frequência, modulos SFP (Small Form Factor Pluggable), microprocessadores, sensores, etc.

6.4 Conclusões

Após o desenvolvimento da placa DETIUA-S3, era muito importante a realização destas interfaces. Com a experiência adquirida no desenvolvimento destas, é possível criar futuramente placas de prototipagem mais complexas, nas quais poderão coexistir vários periféricos.

O desenvolvimento destas placas de interface revelaram uma excelente oportunidade quer para a complementaridade da placa DETIUA-S3 quer para os alunos que as desenvolveram e utilizaram. É de realçar que alguns artigos, que descrevem alguns destes trabalhos desenvolvidos por alunos, vão ser publicados na revista do departamento de electrónica, telecomunicações e informática.

Capítulo 7

Conclusões

Sumário

Este capítulo conclui a tese com a apresentação de um sumário das contribuições principais resultantes do trabalho desenvolvido.

7.1 Trabalho realizado

Esta tese apresenta um caminho que vai desde o desenvolvimento em hardware de um sistema reconfigurável, que utiliza uma FPGA, até à implementação nesse sistema de uma MEF reprogramável.
Globalmente, as contribuições desta tese são:

- Placa de prototipagem DETIUA-S3. Sistema reconfigurável utilizando uma FPGA, que possibilita a sua integração em vários sistemas digitais. A criação desta placa permite ter um sistema reconfigurável, de baixo custo, de fácil manuseamento e de simples compreensão.

- Descrição de métodos e a ferramenta utilizada para a reconfiguração da FPGA através da interface USB e *Bluetooth*. Os métodos implementados para a reconfiguração da FPGA permitem que o sistema reconfigurável seja controlado de uma forma simples e prática, coadjuvado com a ferramenta PBM. Esta apresenta uma interface bastante amigável, o que se torna extremamente útil para utilizadores menos familiarizados nos sistemas reconfiguráveis, mais precisamente, no manuseamento de placas de prototipagem.

- MEF reconfigurável remotamente. A MEF foi baseada em HT, que permite uma

reconfiguração dinâmica parcial de um dispositivo programável estaticamente. A reconfiguração da MEF permite implementar um grande número de interfaces que por sua vez permite resolver um grande número de problemas.

- Criação de módulos/placas de interfaces para comunicação com dispositivos periféricos. O desenvolvimento destes módulos/placas permitiu reduzir de uma forma significativa o valor base da placa de prototipagem. Além disso, permitiu desenvolver o controlo dos vários periféricos, levando a um melhor entendimento no modo de funcionamento dos mesmos.

7.2 Discussão e trabalho futuro

Apesar dos contributos descritos, o trabalho desenvolvido poderia ter ido um pouco mais além, apesar de se considerar que os objectivos propostos foram atingidos. Existem pontos, aos quais se poderia ter dado maior ênfase, mas que poderão ser estudados num trabalho futuro. A seguir são sumariadas algumas ideias cujo conteúdo será interessante explorar.

- Implementação de novos sistemas baseados na placa DETIUA-S3 [11];

- Implementação de uma nova placa de prototipagem, que seria uma versão melhorada da desenvolvida. Esta nova placa poderia integrar outro tipo de interfaces que possibilitariam a integração da mesma em sistemas complexos existentes, tais como, sistemas ópticos e sistemas móveis;

- Criação de um *backplane*, que possibilitaria a integração de várias placas e de versões diferentes. Isto também exigiria métodos e ferramentas diferentes para reconfiguração das FPGAs;

- Estudo mais pormenorizado acerca das MEF e HT, apesar de ser um tema bastante vasto;

- Desenvolvimento de uma mesa de testes digitais para auxílio na análise de circuitos. Esta mesa consistiria num sistema reconfigurável, juntamente com uma zona de placas brancas, que permitiria a concepção e análise de circuitos digitais.

7.3 Publicações

Alguns artigos foram publicados ao longo deste trabalho. Estes permitem observar a evolução do respectivo trabalho e, ao mesmo tempo, verificar se as ideias descritas numa publicação prévia foram, ou não, implementadas posteriormente.
Aqui serão descritos os artigos publicados e as alterações que cada publicação sofreu em relação à anterior.

Os artigos publicados foram:

- Manuel Almeida, "Desenvolvimento de uma Placa Protótipo Baseada numa FPGA", Electrónica e Telecomunicações, Vol. 4, Nº 5, Set. 2005, pp. 656-661.

Este foi o primeiro artigo publicado sobre a placa DETIUA-S3. Neste artigo encontra-se: descrito a arquitectura da placa; apresenta o primeiro protótipo; apresenta a ligação da placa a um dispositivo externo, mais concretamente um LCD; expõe um processador combinatório implementado na placa DETIUA-S3.

- M. Almeida, V. Sklyarov, I. Skliarova, B. Pimentel, "Design Tools for Reconfigurable Embedded Systems", Proceedings of the 2nd International Conference on Embedded Software and Systems – ICESS'2005, IEEE Computer Society, Xi'an, China, December 2005, pp. 254-261.

Este artigo além de conter a informação existente no artigo anterior, apresenta um conjunto de potencialidades futuras, principalmente ao nível da interligação de dispositivos externos.

- Manuel Almeida, Bruno Pimentel, "Ferramentas de Desenvolvimento baseado em Sistemas Reconfiguráveis", Actas da REC2006 – II Jornadas Sobre Sistemas Reconfiguráveis, Porto, Portugal, Fevereiro 2006, pp. 95-98.

Neste artigo é apresentado a primeira versão do PBM onde é possível verificar todas as suas funcionalidades e potencialidades. Além disso é apresentado algum trabalho futuro que viria a ser desenvolvido.

- Manuel Almeida, Bruno Pimentel, "Ferramentas Práticas de Baixo Custo para Desenvolvimento de Sistemas baseados em FPGA's", RESI – Revista Electrónica de Sistemas de Informação, Brasil, 7ª edição, nº1, Julho 2006.

Nesta publicação é apresentada um segundo protótipo da placa DETIUA-S3 onde se encontra implementado, numa placa individual, o módulo da alimentação.

- Manuel Almeida, Bruno Pimentel, Valery Sklyarov, Iouliia Skliarova, "Design Tools for Rapid Prototyping of Embedded Controllers", Proceedings of the 3rd International Conference on Autonomous Robots and Agents – ICARA'2006, Palmerston North, New Zealand, December 2006, pp.683-688.

Esta publicação apresenta pela primeira vez a placa DETIUA-S3 com o módulo *Bluetooth* integrado.

- V. Sklyarov, I. Skliarova, M. Almeida, B. Pimentel, "A Prototyping System for Mobile Devices", Proceedings of the ACM 2007 International Wireless Communications and Mobile Computing Conference – IWCMC'2007, Honolulu, USA, August 2007, pp. 505-510.

São demonstradas algumas potencialidades multimédia da placa. Além disso, também é apresentado um exemplo de MEF reconfigurável e sua aplicação prática.

- V. Sklyarov, I. Skliarova, B. Pimentel, M. Almeida, "Multimedia Tools and Architectures for Hardware/Software Co-Simulation of Reconfigurable Systems", Proceedings of the 21st International Conference on VLSI Design – VLSI Design'2008, Hyderabad, India, January 2008, pp. 85-90.

São descritas várias funcionalidades de simulação *Hardware/Software*, interacção de circuitos com dispositivos físicos e virtuais, e interacção entre placa e utilizadores remotos.

Referências

[1] V. Sklyarov, I. Skliarova, M. Almeida, and B. Pimentel, "A Prototyping System for Mobile Devices," in *Proceedings of the ACM 2007 International Wireless Communications and Mobile Computing Conference - IWCMC'2007*, Honolulu, USA, 2007, pp. 505-510.

[2] V. Sklyarov and I. Skliarova, "Reconfigurable Systems and their Influence on Mobile and Multimedia Applications," in *Proceedings of the 4th International Conference on Advances in Mobile Computing and Multimedia - MoMM'2006*, Tutorial, Ed. Yogyakarta, Indonesia, 2006, pp. 7-8.

[3] K. Parnell and R. Bryner, "Comparing and Contrasting FPGA and Microprocessor System Design and Development," in *WP213*, 2004.

[4] Xilinx Inc. (2004), Revolutionary Architecture for the Next Generation Platform FPGAs. [Online]: http://www.xilinx.com/.

[5] Intel Corporation, Moore's low. [Online]: http://www.intel.com/technology/mooreslaw/index.htm.

[6] R. H. J. M. Otten and P. Stravers, "Challenges in physical chip design," in *Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, San Jose, California 2000, pp. 84 - 91.

[7] M. Devlin, "Multi-FPGA systems for High Performance Computing Applications," UK: Nallatech.

[8] M. Devlin, "Reconfigurable Computing Architectures for High Performance Analysis," UK: Nallatech.

[9] Bellnix, Power Supply Technical Application Note for Xilinx FPGAs. [Online]: http://www.bellnix.com.

[10] Xilinx Inc. (2008), Xilinx Distributor & 3rd Party Boards. [Online]: http://www.xilinx.com/publications/matrix/Dist_ThirdParty_boards.pdf.

[11] V. Sklyarov, I. Skliarova, B. Pimentel, and M. Almeida, "Multimedia Tools and Architectures for Hardware/Software Co-Simulation of Reconfigurable Systems," in *Proceedings of the 21st International Conference on VLSI Design – VLSI Design'2008*, Hyderabad, India, January 2008, pp. 85-90.

[12] Xilinx Inc. (2004), Spartan-3 FPGA Family. [Online]: http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf.

[13] DLP Design, DLP-USB245M User Manual [Online]: http://dlpdesign.com/.

[14] Advanced Micro Devices Inc., AM29LV160D Data Sheet. [Online]: http://www.amd.com/.

[15] M. Almeida and B. Pimentel, "Ferramentas Práticas de Baixo Custo para Desenvolvimento de Sistemas baseados em FPGA's," 7ª ed Brasil: RESI – Revista Electrónica de Sistemas de Informação, 2006.

[16] M. Almeida, B. Pimentel, V. Sklyarov, and I. Skliarova, "Design Tools for Rapid Prototyping of Embedded Controllers," in *Proceedings of the 3rd International Conference on Autonomous Robots and Agents - ICARA'2006*, Palmerston North, New Zealand, 2006, pp. 683-688.

[17] Texas Instruments Incorporated, Power Management. [Online]: http://www.ti.com/.

[18] M. Almeida, V. Sklyarov, I. Skliarova, and B. Pimentel, "Design Tools for Reconfigurable Embedded Systems," in *Proceedings of the 2nd International Conference on Embedded Software and Systems - ICESS'2005*, Xi'an, China, 2005, pp. 254-261.

[19] Cadence Design Systems Inc., Products. [Online]: http://www.cadence.com/products/orcad/index.aspx.

[20] Bluetooth SIG Inc., Bluetooth Technology [Online]: http://www.bluetooth.com/.

[21] M. Almeida and B. Pimentel, "Ferramentas de Desenvolvimento baseado em Sistemas Reconfiguráveis," in *II Jornadas Sobre Sistemas Reconfiguráveis*, Porto, Portugal, 2006.

[22] V. Sklyarov, "Reconfigurable models of finite state machines and their implementation in FPGAs," *Journal of Systems Architecture,* vol. 47, pp. 1043-1064, 2002.

[23] V. Sklyarov, "Hierarchical finite-state machines and their use for digital control," in *IEEE Transactions on VLSI Systems*, 1999, pp. 222-228.

[24] V. Sklyarov, R. Monteiro, and N. Lau, "Integrated Development Environment for Logic Synthesis Based on Dynamically Reconfigurable FPGAs," in *Proceeding of FPL'98*, Tallinn, 1998, pp. 19-28.

[25] V. Sklyarov, "Synthesis of Control Circuits with Dynamically Modifiable Behavior on the Basis of Statically Reconfigurable FPGAs," in *13th Symposium on Integrated Circuits and Systems Design: SBCCI2000*, Manaus, Brazil, 2000, pp. 353-358.

[26] V. Sklyarov, "Synthesis and Implementation of RAM-based Finite State Machines in FPGAs'. Proceedings of FPL'2000," in *Proceedings of FPL'2000*, Villach, Austria, 2000, pp. 718-728.

[27] A. Oliveira, N. Lau, and V. Sklyarov, "Synthesis of VHDL Code from the

Hierarchical Specification of Control Circuits for Dynamically Reconfigurable FPGAs," in *Proceedings of VHDL International User Forum*, Orlando, USA, 1998.

[28] S. Baranov, "Logic Synthesis for Control Automata," in *Kluwer Academic Publishers*, 1994.

[29] J.M.Silva, P.Bjesse, and W.Kunz, "Boolean Satisfiability Solving and its Application in Equivalence and Model Checking," in *ICCAD 2001*, Tutorial, San Jose, 2001.

[30] A.Zakrevskij, "Combinatorial Problems over Logical Matrices in Logic Design and Artificial Intelligence," in *Electrónica e Telecomunicações*, 2 ed. vol. 2, 1998, pp. 261-268.

[31] M.Platzner, "Reconfigurable Accelerators for Combinatorial Problems," in *IEEE Computer*, 2000, pp. 58-60.

[32] J. T. D. Sousa, J. M. Silva, and M. Abramovici, "A Configware/Software Approach to SAT Solving," in *Proc. of the IEEE Symposium on Field-Programmable Custom Computing Machines - FCCM'2001*, 2001.

[33] V.Sklyarov, I.Skliarova, and A.Ferrari, "Hierarchical Specification and Implementation of Combinatorial Algorithms Based on RHS Model," in *Proceedings of the XVI Conference on Design of Circuits and Integrated Systems - DCIS2001*, Porto, 2001, pp. 486-491.

[34] I. Skliarova, "Desenvolvimento de circuitos reconfiguráveis que interagem com um monitor VGA," in *Electrónica e Telecomunicações*. vol. 4, 2005, pp. 626-631.

[35] Adam Chapweske, PS/2 Mouse/Keyboard Protocol. [Online]: http://www.Computer-Engineering.org.

[36] Maxim Integrated Products, Maxim Products. [Online]: http://www.maxim-ic.com/.